Deep Learning Approach to Discontinuity-Preserving Image Registration

by

Eric Ng

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in

The Faculty of Science

Modelling and Computational Science

Ontario Tech University

May 2020

THESIS EXAMINATION INFORMATION

Submitted by: **Eric Ng**

Master of Science in **Modelling and Computational Science**

Thesis title: Deep Learning Approach to Discontinuity-Preserving Image Registration

An oral defense of this thesis took place on May 27, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Lennaert van Veen
Research Supervisor	Dr. Mehran Ebrahimi
Examining Committee Member	Dr. Greg Lewis
External Examiner	Dr. Shahryar Rahnamayan

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Image registration is an indispensable tool in medical image analysis. Traditionally, registration algorithms are aimed at aligning image pairs using regularizers to impose smoothness restrictions on unknown deformation fields. The majority of these methods assume global smoothness in the image domain, which pose issues for scenarios where motion discontinuities exist. Examples where local motion discontinuities happen are the sliding motion between adjacent organ tissues, and the pushing motion of the lungs against the chest wall during the respiratory cycle. Furthermore, an objective function must be optimized for each given pair of images. Thus registration of multiple image sets becomes very time-consuming and poorly scale with higher resolution image volumes. Using recent developments in deep learning, we propose an unsupervised learning-based image registration model. This model is trained over a loss function with a custom regularizer that preserves local discontinuities while simultaneously respecting the smoothness assumption in homogeneous regions of image volumes. In following a learning-based model, the image registration process can be completed within seconds, which is significantly quicker than optimization-based registration algorithms. The proposed model will be evaluated qualitatively and quantitatively on datasets of chest computed tomography (CT) 3D image volumes.

Keywords: image registration, optimization, deep learning, convolutional neural networks, chest CT

Author's Declaration

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I authorize the University of Ontario Institute of Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Statement of Contribution

Part of the work described in Chapter 4 has been published as:

E. Ng, and M. Ebrahimi, "An Unsupervised Learning Approach to Discontinuity-Preserving Image Registration", *in International Workshop for Biomedical Image Registration, WBIR*, Lecture Notes in Computer Science, vol. 12120, pp. 153-162, 2020.

I was responsible for performing the majority of the model design, experiments, and writing of the manuscript.

Acknowledgements

Firstly, I would like to sincerely thank my supervisor Prof. Mehran Ebrahimi for his guidance, advice, and patience throughout this project. He introduced to the world of mathematical imaging since my undergraduate studies, and have since provided me the flexibility and freedom to explore additional topics outside of my thesis subject. For that I will forever be grateful.

I would also like to thank all members of the Modelling and Computational Science program, especially those in Imaging Lab. Not only have you kept me intellectually motivated, but we also built a family-like relationship with one another.

Lastly, and most importantly, I would like to thank my mom. You have seen me at my absolute best and my absolute worst, and not once have you given up on me.

Contents

List of Tables

List of Figures

List of Symbols

$\mathcal{L}, \mathcal{H}, \mathcal{F}$	(Uppercase, stylized) Functionals
x, y, z, f	(Lowercase, italicized) Continuous variables and functions
A, B, C	(Uppercase, italicized) Matrices
$\mathbf{I}, \mathbf{M}, \mathbf{X}, \mathbf{Y}$	(Uppercase, boldfaced) Discrete operators
$\mathbf{x}, \mathbf{y}$	(Lowercase, boldfaced) Vectors
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integers
$f : X \to Y$	Function f maps domain X to codomain Y
$x \in X$	x is a member of the set X
$A \subset B$	The set A is a subset of set B
$\lvert x \rvert$	Absolute value or modulus of x
$\lVert \mathbf{x} \rVert_p$	p-norm of the vector $\mathbf{x}$
$\langle x, y \rangle_V$	Inner product of x and y over the inner product space V
$\lVert x \rVert_V$	Norm of x induced by inner product $\langle \cdot, \cdot \rangle_V$

Chapter 1

Introduction

Common medical imaging techniques include X-ray computed tomography (CT), magnetic resonance imaging (MRI), ultrasound, and positron emission tomography (PET). Unlike taking photos with handheld devices, the luxury of retaking an image does not exist due to its cost and time consuming nature. Consequently, image quality is often sacrificed despite vast improvements in imaging techniques since their inception. To combat these constraints, a large variety of image post-processing techniques have since been developed to help extract meaningful information accurately.

Among the broad field of medical image processing, image registration is the procedure of aligning one or more images with a fixed reference image. Image registration remains as one of the most commonly researched subfields due to its vast application both inside and outside the medical domain such as motion prediction, target tracking, image stitching, and satellite image alignment. When multiple images of the same scene are taken, misalignment of images becomes unavoidable due to a variety of factors such as camera positioning, distortion of sensors, and natural movement. In medical imaging, image registration is an indispensible tool as it can be used to account for natural movements of internal organs (e.g. cardiac and respiratory cycles), as well as tracking and predicting growth of malignant tumors within cancer patients.

Image registration involves determining a transformation that morphs one image to another. The problem itself is naturally ill-posed, which means there may be more than one solution. Many existing work address this by restricting transformations based on material properties of the objects captured within the images. However, these restrictions are often imposed globally on all regions of the images, which do not apply to image regions containing organ boundaries. As a result, additional care must be considered for these regions, especially at the interface between two different materials (for instance, brain matter/skull interface, and lung/spine interface). In these cases, one expects that motion on opposite sides of the interface are noticeably different, i.e. a discontinuity in the displacement field.

The work presented in this thesis focuses on the discontinuity-preserving image registration problem. In addition, we take advantage of the recent advances in deep learning. In this work, we train an artificial neural network that takes in two images as inputs to predict the displacement field that morphs one image to the other. To detect and enforce local image discontinuities, a custom regularizer is designed by dividing image volumes into smaller image patches, then compares local displacement vectors with their neighboring vectors. We evaluate our proposed model on datasets provided by DIR-Lab [5, 6] and the POPI-model [7]. By taking a learning-based approach, our model has an additional benefit where registration time is reduced by up to 100 times compared to classical methods.

1.1 Thesis Outline

The remainder of this thesis is organized as follows.

In Chapter 2, we explore the mathematics behind the image registration problem as an ill-posed inverse problem. Standard metrics that measure image similarity and dissimilarity are discussed, as well as commonly used regularizers in a classical image

registration framework. As image registration is classically framed as an energy minimization problem, the optimize-then-discretize and discretize-then-optimize approaches will be reviewed.

In Chapter 3, we explore the field of deep learning, a popular subfield of modern machine learning. We first formally define supervised and unsupervised learning. The mathematical construct behind artificial neural networks are reviewed. We discuss optimization techniques, notably backpropagation, that are used during the training step of a neural network.

Chapter 4 first presents the motivation behind the discontinuity-preserving image registration problem. We then proceed to present a learning-based model to tackle the problem. A custom regularizer is designed to address motion discontinuities which is used, in part, to train the artificial network. Additionally, an efficient implementation of the loss function is presented, which reduces memory requirements by several times compared to a naive implementation. Experimental results and comparisons with existing methods are also presented in this chapter.

Chapter 5 discusses the strengths and weaknesses of the proposed model. Ideas to address these weaknesses are presented which can be used to steer the direction towards future research.

1.2 Software and Source Code

Software

The implementation and evaluation of the model presented in this thesis are written in Python. Python was designed with simple code readability in mind in order to reduce complexity required to maintain and update existing code. Although it is not as fast as lower level languages like Fortran and C, the latest advances in computer hardware have considerably shrunk the difference in computation time. Python have since expanded

into a language that is capable of computing and analyzing large quantities of data, in addition to becoming the staple for modern machine learning. For the experiments in this thesis, the majority of computations are performed on graphical processing units (GPUs), while central processing units (CPUs) are reserved for data loading and pre-processing purposes. The following Python packages are used in this project.

- **NumPy** is a library created for the purpose of scientific computing in Python. Its design is highly similar to MATLAB, containing a large collection of mathematical functions and operations that can be applied to data stored as multi-dimensional arrays.
  `https://numpy.org/`

- **Scikit-image** is an extension of the larger SciPy library which contains algorithms used specifically for image processing tasks.
  `https://scikit-image.org`

- **Pydicom** is used to read and load images in DICOM format.
  `https://pydicom.github.io/`

- **PyTorch** is an open source deep learning library that is syntactically simialr to NumPy. Like many existing machine learning libraries, PyTorch supports GPU acceleration through Nvidia's CUDA, where data is stored in multidimensional arrays called tensors. More prominently, PyTorch contains autograd [8], a subpackage that enables automatic differentiation for all tensor-related operations.
  `https://pytorch.org/`

Source Code

The implementation of our model will be made available in the following link:
`https://github.com/eshn/DPIR`

Chapter 2

Image Registration

The process of image registration requires finding a transformation such that the moving image under such transformation is visually similar to the fixed image [9]. The type of transformation allowed is either pre-determined (parametric transformation) or controlled using a regularizer that computes a penalty score based on some property of the transformation (deformable/free-form transformation). However, it is possible to define regularizers that yield parametric transformations as ideal solutions, such as the thin-plate spline [10]. The image registration problem is then framed as an optimization problem by minimizing some objective function using one of two approaches: discretize-then-optimize, or optimize-then-discretize.

In the discretize-then-optimize approach, the continuous image domain is converted to a discrete domain. Image data can then be viewed as a vector in a finite dimensional vector space instead of a function in some function space (Banach or Hilbert space). The problem can then be optimized with a variety of a techniques; from simple algorithms such as gradient descent, to advanced methods such as Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [11]. Conversely, the optimize-then-discretize approach treats image data as a scalar function in a continuous domain. Optimization is then presented as a variational problem, leading to the Euler-Lagrange equation that is solved numerically.

The solution, under either approach, yields a transformation that may potentially map grid locations to non-grid locations. Interpolation then becomes inevitable in order to produce the transformed image for visual comparisons.

In summary, a classical image registration algorithm consists of the four key components: 1. Penalty function (distance measure and regularization); 2. Transformation model (parametric, or free-form); 3. Optimization algorithm, and; 4. Interpolation.

2.1 Introduction

We first establish the mathematical language used in the traditional image registration problem. Consider a reference image $\mathcal{R}$ and a template image $\mathcal{T}$ (sometimes known as fixed and moving images respectively). Images can be interpreted as mappings $\mathcal{R} : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ and $\mathcal{T} : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ where Ω is the image domain and d is the dimensionality of the image. Intuitively, an image is simply a function that maps a location in d-dimensional space to the intensity value at that particular location known as a pixel (or more generally, a voxel). Image intensities are represented as rank-d tensors during numerical implementation. In this thesis, our focus will be on the 3-dimensional case ($d = 3$), however, the methods can be generalized to any number of dimensions given that computational constraints are met.

The image registration problem requires finding a transformation $y : \mathbb{R}^3 \to \mathbb{R}^3$ such that the template image transformed under y is "similar" to the reference image. An alternative formulation is to decompose the transformation as $y(x) = x + u(x)$: a sum of the original image coordinates and some displacement field $u \in L^2(\mathbb{R}^3)$, where $L^2(\mathbb{R}^3)$ is set of square-integrable[1] functions, in the Lebesgue sense, in $\mathbb{R}^3$. Numerical algorithms are implemented to either maximize image similarity or minimize image dissimilarity.

[1]Strictly speaking, elements of $L^2(\mathbb{R}^3)$ are equivalence classes of square-integrable functions in $\mathbb{R}^3$. Two functions f and g are in the same equivalence class if the support of $(f - g)$ has measure zero.

Naturally, image registration is designed as an energy minimization problem in the form

$$\mathcal{J}[u] := \mathcal{D}[\mathcal{T}(x + u(x)), \mathcal{R}(x)] + \alpha\mathcal{S}[u] \tag{2.1}$$

where $x \in \Omega$. Here, $\mathcal{D}$ is a distance measure, or the data fidelity term, that quantifies the similarity/dissimilarity between image pairs. Ideally, we would like the problem to be *well-posed*, where a solution exists, is unique, and depends continuously on the data [12]. However, this is often not the case as there may be multiple, and often infinite, number of possible solutions. An example is shown in Figure (2.1), where the transformation can take on many possibilities such as horizontal/vertical reflection and rotation about the center by $(k + 1)\pi/2$, where $k \in \mathbb{Z}$. The image registration problem is therefore *ill-posed*, thus a regularizer $\mathcal{S}$ is necessary to reduce the amount of candidate solutions in the solution space. The scalar $\alpha > 0$ serves as a weight parameter that controls the regularizer's impact.

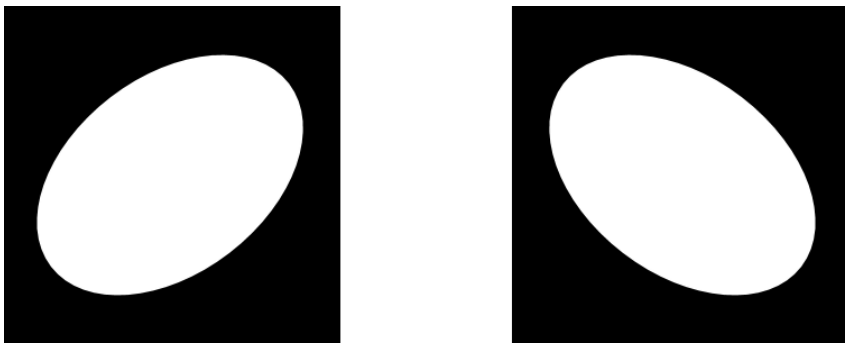

Figure 2.1: Toy example of ill-posedness in the image registration problem. Possible transformation include horizontal/vertical reflection and rotation by $(k + 1)\pi/2$, where $k \in \mathbb{Z}$.

2.1.1 Distance Measures

Distance measures are necessary tools to validate visual consistency between image pairs. In this section, we explore various distance measures commonly used in image registration and discuss their respective strengths and shortcomings.

Mean Absolute Error Mean absolute error (MAE) is a metric frequently used in many fields such as statistics, information theory, and signal processing. Given two vectors $\vec{x} = (x_1, \ldots, x_n)^T, \vec{y} = (y_1, \ldots, y_n)^T$, MAE of $\vec{x}$ and $\vec{y}$ is defined as

$$\mathrm{MAE}(\vec{x}, \vec{y}) := \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \,. \tag{2.2}$$

This definition can be easily extended to 2D and 3D image volumes by simply enumerating pixel/voxel locations to corresponding indices. More generally, MAE can be viewed as the distance function

$$d(\vec{x}, \vec{y}) = \frac{1}{n} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

where $p = 1$. Under this formulation, MAE is simply a specific case of a family of ℓ_p-norm based distance measures. Choosing $p = 2$ corresponds to another well-known metric: root mean squared error (RMSE) which is equivalent to the standard deviation of residual errors.

MAE (and its close relatives) is extremely popular due to its simplicity and low computational requirement ($O(n)$ where n is the length of the input vector). However, it suffers from significant drawbacks. MAE computes the average of the absolute error at each matching index, which is equivalent to measuring pixel-wise error between images. As the error is captured on a pixel-level, MAE completely neglects neighboring information, hence it is unable to capture image features that is otherwise obvious to the human eye. For instance, consider two images where one image is a translation of the other by a single pixel. MAE will penalize the two images heavily despite the two images appearing nearly identical. Furthermore, since MAE relies on direct comparisons between intensity values, these values must be measured using a consistent method, thus image pairs must share the same modality. Depending on the application, this may not be practical in a medical imaging scenario as the involved images may be captured using different techniques, such as computed tomography (CT) and magnetic resonance imaging (MRI).

Normalized Cross-Correlation Cross-correlation measure the similarity of two signals based on the *translation* of one signal with another. In imaging applications, the translation is typically removed from cross-correlation as it involves large regions where image data is undefined. Normalized cross-correlation (NCC) restricts the upper bound to 1 as cross-correlation is potentially unbounded prior to normalization. Mathematically, NCC is defined as

$$\mathrm{NCC}(\vec{x}, \vec{y}) := \frac{\langle \vec{x}, \vec{y} \rangle^2}{\|\vec{x}\|^2 \|\vec{y}\|^2} \tag{2.3}$$

where $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ are the Euclidean inner product and norm respectively. For imaging related tasks, image intensity values are stored in $\vec{x}$ and $\vec{y}$. Since NCC is maximized when $\vec{x} = \vec{y}$, it is often re-formulated by minimizing $1 - \mathrm{NCC}$ instead. The negative sign transforms the maximization problem into a minimization problem, and 1 is added to ensure that $0 \leq 1 - \mathrm{NCC} \leq 1$. In many applications, a small constant $\epsilon > 0$ is added to the denominator to prevent from dividing by zero. This is omitted as the only scenario where this occurs whenever either $\vec{x}$ or $\vec{y}$ is the zero vector, which corresponds to a pitch black image.

Normalized Gradient Field Rather than determining image similarity/dissimilarity using pixel intensity values directly, normalized gradient field (NGF) measures similarity by quantifying how intensity values change across two images. Intuitively, if two images are similar, then their corresponding intensity changes must also be similar at the same location. The gradient field is normalized so that the location of the change is emphasized over the strength of the change. NGF [13, 14] is defined as

$$\mathrm{NGF}(\vec{x}, \vec{y}) := 1 - \mathrm{n}[\vec{x}]^T \mathrm{n}[\vec{y}] \tag{2.4}$$

where

$$\mathrm{n}[\vec{x}; \varepsilon] = \frac{\nabla \vec{x}}{\sqrt{\|\nabla \vec{x}\|^2 + \varepsilon^2}}. \tag{2.5}$$

The constant $\varepsilon \geq 0$ is a noise threshold parameter that serves two purposes: 1. To resolve the non-differentiability issue at locations where the gradient is zero, and; 2. To

differentiate edges ($\|\nabla\vec{x}\| > \varepsilon$) from image noise ($\|\nabla\vec{x}\| < \varepsilon$). This is a necessary step since noise have large local gradients, which can be mistaken for edge information if left unchecked. The choice of ε essentially answers the question "how small can image gradient be before it is considered as noise?". One proposed choice is the normalized total variation (TV) [13]

$$\varepsilon = \frac{\eta}{V}\int_{\Omega}\|\nabla I(x)\|dx \tag{2.6}$$

where ∇I is the gradient of image intensity values, $\eta \geq 0$ is a noise estimate parameter, and V is the volume of the image domain Ω. In numerical implementations, the integral is replaced with summations over the spatial dimensions and the gradient is approximated as

$$\nabla I \approx (\partial_1^h I, \partial_2^h I, \ldots, \partial_d^h I) \tag{2.7}$$

where ∂_k^h is a finite difference approximation in the k'th spatial direction under homogeneous Neumann boundary conditions of step size h.

2.1.2 Regularization

One of the most difficult challenges of image registration is due to the *ill-posed* nature of the problem. Many methods have been introduced to remove unfeasible solutions from the entire solution space. This is most commonly achieved by imposing constraints via the use of regularizers such as diffusion [15] and elastic [16]. Alternatively, one may assume the transformation model takes on a parametric form such as affine transformation. As a middle ground, it is entirely possible to meet halfway: by deisgning regularizers that yield minimizers in a parametric form. An example of this is Thin Plate Splines (TPS) by Duchon [17], where the optimal solution is the sum of a linear combination of radial basis functions and an affine transformation. In this section, we will explore several regularization methods most commonly used in modern image registration algorithms.

Derivative-Based Regularization Most regularizers in traditional image registration literature are derivative-based. Since the derivative explains rate of change, using derivatives to construct regularizers effectively controls the smoothness of the transformation. Derivative-based regularizers are typically in the form

$$\mathcal{S}[u] = \int_\Omega \|\mathcal{B}[u]\|_2^2 dx \tag{2.8}$$

where $\mathcal{B}$ is a differential operator that defines the regularizer, and $\|\cdot\|_2$ denotes the Euclidean norm. For instance, the diffusion regularizer is

$$\mathcal{S}_{diff}[u] = \int_\Omega \sum_{i=1}^{d} \|\nabla u_i\|^2 dx \tag{2.9}$$

with $u = (u_1, \ldots, u_d)^T$. Diffusion regularizer essentially quantifies variations of u and serves as the heart of many optical flow problems such as object tracking [15].

The elastic regularizer is another regularization choice used frequently in registration where the image content is assumed to behave like an elastic material by penalizing the elastic potential energy required to "deform" the image. The elastic regularizer is given by

$$\mathcal{S}_{elastic}[u] = \int_\Omega \mu\langle \nabla u, \nabla u\rangle + (\lambda + \mu)(\nabla \cdot u)^2 dx \tag{2.10}$$

where μ and λ are Lamé constants in continuum mechanics [18]. The constant $\mu > 0$ is the shear modulus of an elastic material, and is related to λ via the relationship $\lambda = K - \frac{2}{3}\mu$ where K is the bulk modulus. Since the regularizer is physically motivated, it has become one of the most commonly used regularization tools for image registration problems.

A regularizer based on second order derivatives is the curvature regularizer defined as

$$\mathcal{S}_{curv}[u] = \int_\Omega \sum_{i=1}^{d} (\Delta u_i)^2 dx \tag{2.11}$$

where $\Delta = \nabla^2$ is the Laplacian operator. Note that the minimizer of the curvature regularizer must satisfy $\Delta u_i = 0$ for all i, i.e. a harmonic vector field. Thus curvature

regularizers produce results that are often smoother than the ones obtained from first-order regularizers like diffusion and elastic regularizers.

For a more comprehensive study of derivative-based regularizers and their numerical implementations, we refer to the book *FAIR: Flexible Algorithms for Image Registration* by Modersitzki [14].

Thin Plate Spline Thin plate spline (TPS) is a spline method that simulates the bending energy required to deform a thin sheet of metal. TPS is used in landmark-based image registration, where *landmarks*, or *control points* are selected in the reference and template images manually prior to registration. Through these landmarks, they provide an additional point correspondant constraint which transforms the image registration problem into an interpolation problem. Let $\{r_j\}, \{t_j\}, j = 1, \ldots, n$ be two sets of unique control points in the reference and template images respectively, then the interpolating function $u : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ satisfying

$$u(t_j) = r_j, \quad j = 1, \ldots, n \tag{2.12}$$

must be a minimizer of the functional

$$\mathcal{M}[u] = \sum_{j=1}^{n} \|u(t_j) - r_j\|^2. \tag{2.13}$$

In this formulation, there is no restriction on u hence it can be a piecewise linear function. A polyharmonic regularizer defined as

$$\mathcal{R}[u] = \int_{\Omega} |\nabla^m u|^2 dx \tag{2.14}$$

to impose smoothness. Thus the objective functional becomes

$$\mathcal{J}[u] = \sum_{j=1}^{n} \|u(t_j) - r_j\|^2 + \lambda \int_{\Omega} |\nabla^m u|^2 dx. \tag{2.15}$$

The minimizer of Equation (2.15) is called the polyharmonic spline. Choosing $m = 2$, the polyharmonic operator becomes the biharmonic operator, and the interpolant is the

thin plate spline. Under this formulation, Wahba [19] guaranteed the existence of a unique minimizer. Furthermore, the unique minimizer has a closed-form representation composed of an affine part and a non-affine part in the form

$$u(x) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + \sum_{j=1}^{n} c_j \varphi(\|x - t_j\|_2) \tag{2.16}$$

where $\varphi(r)$ is the fundamental solution of the biharmonic operator, i.e. the solution to $\Delta^2 \varphi = \delta(x)$, and $\delta(x)$ is the Dirac delta distribution centered around the origin. For odd and even dimensions, φ take the forms $\varphi(r) = r$ and $\varphi(r) = r^2 \log r$ respectively. Geometrically, the affine part of the minimizer accounts for large-scale global movement such as translation and rotations, and the non-affine part characterizes local deformations such that the interpolation conditions are not violated. The set of φ is often referred to as radial basis functions (RBF) since each φ is radially symmetric and centered around each control point.

This begs the question, how are the coefficients recovered. Since this is ultimately an interpolation problem, then the following must be true for all $j = 1, \ldots, n$:

$$r_j = u(t_j) = a_0 + a_1 t_j^1 + a_2 t_j^2 + a_3 t_j^3 + \sum_{i=1}^{n} c_i \varphi(\|t_j - t_i\|_2). \tag{2.17}$$

To express this more compactly, let us define the matrix $\mathbf{A}$ where its entry in the i'th row and j'th column is $A_{ij} = \varphi(\|t_i - t_j\|_2)$. Also let

$$\mathbf{B} = \begin{bmatrix} 1 & t_1^1 & t_1^2 \\ 1 & t_2^1 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_n^1 & t_n^2 \end{bmatrix}, \tag{2.18}$$

$\mathbf{c} = (c_1, \ldots, c_n)^T$, and $\mathbf{a} = (a_0, a_1, a_2, a_3)^T$, and $\mathbf{r} = (t_1, \ldots, t_n)^T$, then the interpolating conditions can be reduced to the block linear system

$$\begin{bmatrix} \mathbf{A} + \lambda \mathbf{I} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0} \end{bmatrix}. \tag{2.19}$$

The block matrix has a trivial nullspace, thus guaranteeing a unique solution of coefficients. Furthermore, the block matrix is symmetric, hence the system can be solved efficiently using Cholesky decomposition followed by forward/backward substitution. For a detailed proof on the invertibility of this system, we refer to the book *A Course in Approximation Theory* [20].

Reproducing Kernel Hilbert Space Although thin plate spline produces a unique interpolant with an explicit closed form expression, it is restricted to the biharmonic operator as regularization. Kernel methods generalize the idea of finding a minimizer that has a parametric representation based on carefully designed regularizers. In particular, solutions from reproducing kernel hilbert spaces (RKHS) were popularized in classical machine learning and information theory [21, 22], and have recently been applied to image registration problems [3, 23, 24, 25].

Definition 2.1.1 (Kernel)**.** *Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a **kernel** if there exists a real Hilbert space $\mathcal{H}$ and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ such that*

$$k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$$

for all $x, y \in \mathcal{X}$

In essence, the function ϕ defines some representation for elements of $\mathcal{X}$ in a Hilbert space $\mathcal{H}$, which can be interpreted as encoding data into a higher dimensional (potentially infinite) representation. Inner products between representations can then be performed by a simple function evaluation of the original data. Since no restrictions are placed on $\mathcal{X}$, its elements can be non-mathematical objects like images, cars, and people. An important property of kernels is that they are positive definite, since

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle a_i \phi(x_i), a_j \phi(x_j) \rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^{n} a_i \phi(x_i) \right\|_{\mathcal{H}}^2 \geq 0 \end{aligned} \tag{2.20}$$

with equality holding if and only if $a_i = 0$ for all $i = 1, \ldots, n$. Consider now the evaluation functional $\mathcal{L}_x : \mathcal{H} \to \mathbb{R}$ that evaluates *any* function in $\mathcal{H}$ at x, i.e. $\mathcal{L}_x[f] = f(x)$ for some $f \in \mathcal{H}$. It is evident that $\mathcal{L}_x$ is linear since

$$\mathcal{L}_x[f+g] = (f+g)(x) = f(x) + g(x) = \mathcal{L}_x[f] + \mathcal{L}_x[g], \tag{2.21}$$

$$\mathcal{L}_x[\alpha f] = \alpha f(x) = \alpha \mathcal{L}_x[f]. \tag{2.22}$$

Furthermore, $\mathcal{L}_x$ is continuous. Therefore $\mathcal{L}_x$ is a bounded operator on $\mathcal{H}$. By the Riesz Representation Theorem [26], there exists $g \in \mathcal{H}$ such that $\mathcal{L}_x[f] = \langle f, g \rangle$ for all $f \in \mathcal{H}$. Suppose we define $K_x := g$, then

$$f(x) = \mathcal{L}_x[f] = \langle f, g \rangle = \langle f, K_x \rangle. \tag{2.23}$$

This is referred to the *reproducing property*, or informally, the *kernel trick*. A second evaluation functional $\mathcal{L}_y$ can be defined in a similar fashion. Invoking Riesz Representation Theorem a second time shows that

$$K_x(y) = \mathcal{L}_y[K_x] = \langle K_x, h \rangle = \langle K_x, K_y \rangle \tag{2.24}$$

for some $h =: K_y \in \mathcal{H}$. which quickly leads to

$$\mathcal{L}_y[K_x] = K_x[y] = \langle K_x, K_y \rangle. \tag{2.25}$$

The reproducing kernel is thus defined as

$$k(x, y) = \langle K_x, K_y \rangle. \tag{2.26}$$

A reproducing kernel hilbert space is a Hilbert space where such reproducing kernel is well-defined. Since $\mathcal{H}$ is potentially infinite-dimensional, this allows any inner products of $\mathcal{H}$ to be computed via a simple function evaluation, which drastically reduces the computational time required.

Theorem 2.1.1 (Representer Theorem)**.** *Let $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n \in \mathcal{X}^n$. Consider the minimization problem*

$$\arg\min_{f \in \mathcal{H}} E[x_i, y_i; f] + g(\|f\|_{\mathcal{H}})$$

where $g(\cdot)$ is a strictly increasing function and E is an empirical loss function, then its minimizer f^* is in the form

$$f^*(\cdot) = \sum_{i=1}^{n} c_i k(x_i, \cdot)$$

where c_i are constants.

Proof. See Appendix (A). □

Since the minimizer must be a finite linear combination of kernel functions, this reduces the dimensionality of the problem to n, the number of data points. Two examples of kernel functions are Gaussian kernel and the Taylor series kernel defined respectively in equations (2.27) and (2.28) as:

$$k_1(x, y) = \exp\left(-\gamma^{-2}\|x - y\|^2\right), \tag{2.27}$$

$$k_2(x, y) = \sum_{n=0}^{\infty} a_n(\langle x, y\rangle)^n. \tag{2.28}$$

One particular family of kernel functions of interest are the *Wendland kernels* [27], which have compact support and are differentiable up to the user's choice. Since these kernels evaluate to zero outside of some closed interval, gradient approximations also evaluate to zero outside of its support which results in a sparse linear system. Thus RKHS methods are ideal for optimization problems solved with gradient-based algorithms. The Wendland kernels for $d = 2$ and $d = 3$ are outlined in Table (2.1). RKHS methods have

Kernal name	**Kernel function**
Wendland C^0	$\psi_{2,0} = (1 - r)_+^2$
Wendland C^2	$\psi_{3,1} = (1 - r)_+^4(4r + 1)$
Wendland C^4	$\psi_{4,2} = (1 - r)_+^6(35r^2 + 18r + 3)$

Table 2.1: Wendland kernels in 2D and 3D. $r_+ = \max(0, r)$.

demonstrated remarkable success in image registration problems in recent years [3, 25, 24] by formulating regularizers satisfying the criterion imposed by the representer theorem.

The extended representer theorem by Jud *et al.* [23] further generalizes the representer theorem, which shows that objective functionals in the form

$$E[x_i, y_i; f] + \sum_{i=1}^{L} g_i(p_i(f_0) + \|v_f\|_{\mathcal{H}}) \tag{2.29}$$

also have minimizers in their prescribed RKHS.

2.2 Optimization

As discussed in the previous section, image registration is typically modelled as an energy minimization problem in the form

$$\mathcal{J}[u] = \mathcal{D}[\mathcal{T}(x + u(x)) + \mathcal{R}(x)] + \alpha\mathcal{S}[u]. \tag{2.30}$$

Since the image registration problem is an inverse problem by design, optimization is an essential step in the registration models. Optimization methods can be classified into two streams: optimize-then-discretize and discretize-then-optimize.

2.2.1 Optimize-then-Discretize

In the **optimize-then-discretize** approach, the objective functional is kept in its continuous form. Similar to the standard derivative in elementary calculus, the objective functional $\mathcal{J}$ is minimized whenever its Gâteaux derivative, or functional/variational derivative, equals zero:

$$d\mathcal{J}[u; v] = \lim_{\tau \to 0} \frac{\mathcal{J}[u + \tau v] - \mathcal{J}[u]}{\tau} = 0 \tag{2.31}$$

where v is an arbitrary function that is differentiable in its support Ω. Taking the Gâteaux derivative of equation (2.30) yields

$$d\mathcal{J}[u; v] = d\mathcal{D}[u; v] + \alpha d\mathcal{S}[u; v] = 0 \tag{2.32}$$

known as the **Euler-Lagrange equation** [28]. The above can be rewritten as

$$d\mathcal{S}[u; v] = -\alpha^{-1} d\mathcal{D}[u; v], \tag{2.33}$$

or equivalently,

$$\mathcal{A}[u] = \alpha^{-1} f(x, u, \mathcal{T}, \mathcal{R}) \tag{2.34}$$

where $\mathcal{A}$ is a partial differential operator that arises from the Gâteaux derivative of the regularizer $\mathcal{S}$. For instance, the diffusion regularizer $\mathcal{S}_{\text{diff}}[u]$ corresponds to the negative Laplacian operator

$$\mathcal{A}_{\text{diff}}[u] = -\Delta u, \tag{2.35}$$

and the curvature regularizer $\mathcal{S}_{\text{curv}}[u]$ corresponds to the biharmonic operator

$$\mathcal{A}_{\text{curv}}[u] = \Delta^2 u. \tag{2.36}$$

On the right hand side of the equation, f acts as a "force" vector field induced by the chosen distance measure. Finding the analytic solution of equation (2.34) is often too time-consuming, or in many cases, simply not possible. Thus it is solved numerically by introducing an artificial time variable t, then modifying the Euler-Lagrange equation to be in the form

$$\partial_t u + \mathcal{A}[u] = \alpha^{-1} f(x, u, \mathcal{T}, \mathcal{R}) \tag{2.37}$$

where ∂_t denotes the partial derivative with respect to t. Solving equation (2.34) is now equivalent to finding the steady state solution of equation (2.37). The spatial domain is then discretized in a grid-like manner as most imaging applications involve rectangular domains. The displacement field u can now be represented by a $(d+1)$-dimensional array of size $n_1 \times \cdots \times n_d \times d$, where n_i is the number of discretization points in the i'th direction, and the last dimension stores each coordinate component of the displacement field. The array is then stored in lexicographic ordering, which transforms the discrete displacement field $\mathbf{u}$ into a column vector by concatenating all dimensions of the array against the first dimension as seen in Figure (2.2). $\mathcal{A}$ can then be approximated by a

sparse matrix A during discretization. The fully semi-implicit discretized model then becomes

$$\frac{\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}}{t} + A\mathbf{u}^{(k+1)} = \alpha^{-1}\mathbf{f}(\mathbf{x}, \mathbf{u}^{(k)}, T, R) \tag{2.38}$$

where t is the step size in time, and $\mathbf{u}^{(k)}$ is a discrete approximation of the displacement field at time step k. The above can be rearranged algebraically to

$$\mathbf{u}^{(k+1)} = (\mathrm{Id} + tA)^{-1}\left[t\alpha^{-1}\mathbf{f}(\mathbf{x}, \mathbf{u}^{(k)}, T, R) + \mathbf{u}^{(k)}\right] \tag{2.39}$$

which can be written more compactly as

$$\mathbf{u}^{(k+1)} = \boldsymbol{F}(\mathbf{u}^{(k)}) \tag{2.40}$$

for a corresponding operator $\boldsymbol{F}$. $\boldsymbol{F}$ is made a contractive map by choosing an appropriate step size t such that the Lipschitz constant of $\boldsymbol{F}$ is less than one. Thus repeated applications of equation (2.39) will generate a sequence that converges to a fixed point via the contractive mapping theorem. Like many iterative schemes, the final solution is highly sensitive to its initial conditions. In medical image registration, it is assumed that image displacements are small. Therefore the identity transformation is frequently chosen as the initial transformation. This corresponds to the zero displacement field, i.e. $\mathbf{u}^{(0)} = \mathbf{0}$.

2.2.2 Discretize-then-Optimize

The **discretize-then-optimize** approach first discretizing the objective functional using finite differencing and Riemann sums to approximate any derivatives and integrals:

$$J(\mathbf{u}) = D(T(\mathbf{x} + \mathbf{u}(\mathbf{x})), R(\mathbf{x})) + \alpha S(\mathbf{u}) \tag{2.41}$$

where D and S are the discrete counterparts of the distance measure $\mathcal{D}$ and regularizer $\mathcal{S}$ respectively. The minimizer is found by searching for $\mathbf{u}^*$ such that $\nabla_{\mathbf{u}} J(\mathbf{u}^*) = \mathbf{0}$, using a derivative-based algorithm such as *gradient descent*,

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - \gamma \nabla J(\mathbf{u}^{(k)}). \tag{2.42}$$

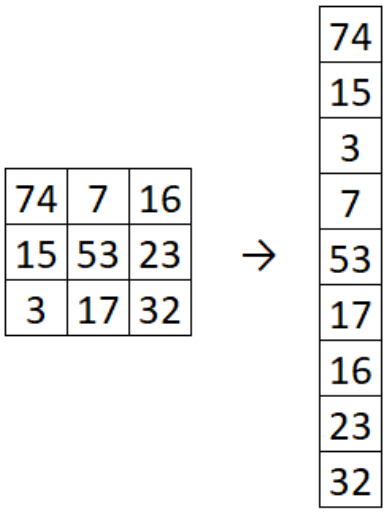

Figure 2.2: Example of lexicographic ordering for a 2D scalar field.

Gradient descent searches for the direction with the steepest slope (the direction with largest directional derivative), and takes a step in the opposite direction by a step-size γ. In the cases where the step is taken in the direction of the positive gradient, the algorithm is known as *gradient ascent* which is used in maximization problems. Since gradient descent relies solely on first derivatives, it is considered a *first-order optimization algorithm*. In many cases, convergence can be improved by including, or approximating, second derivative information. A well-known example of a *second-order optimization algorithm* is the Newton-Raphson method. In particular, consider the optimization problem

$$\min_{\mathbf{u}} \mathcal{L}(\mathbf{u}) \tag{2.43}$$

where $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is a twice-differentiable scalar function. This optimization problem is transformed into a root-finding problem as $\nabla_{\mathbf{u}}\mathcal{L} = \mathbf{0}$ is a necessary condition for optimality. Thus the root can be approximated iteratively via the update

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - [H(\mathcal{L})(\mathbf{u}^{(k)})]^{-1}\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}^{(k)}). \tag{2.44}$$

Note that this requires not only the Hessian matrix H, but also its inverse at each iteration. Rather than computing the matrix inverse, let $\mathbf{h}^{(k)} = [H(\mathcal{L})(\mathbf{u}^{(k)})]^{-1}\nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}^{(k)})$.

Then $\mathbf{h}^{(k)}$ can be computed by solving the linear system

$$[H(\mathcal{L})(\mathbf{u})^{(k)}]\mathbf{h}^{(k)} = \nabla_{\mathbf{u}}\mathcal{L}(\mathbf{u}^{(k)}). \tag{2.45}$$

H is approximated at each time step rather than fully computed. Any algorithm where the Hessian matrix is approximated are known as Quasi-Newton methods, such as Broyden-Fletcher-Goldbarb-Shanno (BFGS) and Symmetric Rank 1 (SR1). For an in depth study of Quasi-Newton methods, we refer to the book *Numerical Optimization* by *Jorge Nocedal* and *Stephen J. Wright* [29].

In most cases, the objective function is not convex, therefore there is a strong possibility that the minimizer found is a local minimum. A *multi-scale* approach increases the possibility of obtaining a global minimizer. The minimization problem is first solved on a coarse grid, where the solution is likely a global minimizer at that particular discretization scale. The domain is re-discretized on a finer scale, and the minimization problem is solved again using the solution from the previous scale as its initial condition. This process is repeated until the discretization scale is reached [30].

2.3 Summary

In this chapter, we presented an overview of the classical image registration problem, which requires finding a transformation $y = x + u(x)$ such that the transformed moving template image is "similar" to the fixed reference image. Transformations are classified into two primary types: parametric and free-form. In parametric registration, a parametric transformation model such as rigid and affine transformation is assumed. This approach significantly reduces the degrees of freedom but is very restrictive in terms of the types of transformations allowed. On the other hand, free-form deformations allow larger varieties of transformations, where unfeasible ones are eliminated through the use of regularizers. Thus, the objective functional can be summarized into two parts: a distance measure that quantifies image similarity and/or dissimilarity, and regularization.

We further explored that models such as thin plate spline and RKHS models combined these two idea, by carefully defining regularizers that yield parametric minimizers.

Next, we focused on the two optimization paradigms: optimize-then-discretize and discretize-then-optimize. In the optimize-then-discretize framework, the Gâteaux derivative of the objective functional is computed and set to zero. This gives the Euler-Lagrange equation, an inhomogeneous partial differential equation which is solved numerically. Conversely, the discretize-then-optimize approach performs discretization in its initial step, transforming the objective functional into a multivariate scalar function, which is then optimized using a derivative-based optimization algorithm such as gradient descent.

For imaging tasks, machine learning algorithms can be broadly viewed as optimization problems in disguise that follow the discretize-then-optimize paradigm. The basics of these algorithms will be discussed in the upcoming chapter.

Chapter 3

Deep Learning

In this chapter, we will explore the basics of deep learning, a branch of machine learning. Invented by Arthur Samuel [31], machine learning is defined as a field of study that gives computers the ability to learn without being explicitly programmed. Similar to humans, learning requires exposure to very large quantities of previous data in addition to strong computing power. These two resources were simply not available in the 1950s. Over the past ten years, the availability of open data and computational power have exploded, which transformed what was once fantasy into reality.

From a mathematical perspective, machine learning involves finding a mapping $f : X \to Y$ where X and Y are input and output spaces respectively. The function f is parameterized by a vector of trainable parameters θ, thus learning is achieved by finding optimal values of θ that most accurately reflects the relationship between X and Y.

In this chapter, we will begin by discussing the two broad categories of machine learning algorithms: supervised and unsupervised learning. We will continue into artificial neural networks which is the staple of modern machine learning. Finally, we will then discuss the steps required to train a model.

3.1 Supervised versus Unsupervised Learning

Supervised Learning In the most general sense, the purpose of machine learning is to predict a specific outcome based on a given input. Depending on the data provided, the true outcome, also known as *labels*, may or may not be available. In cases where labels are available, they can be directly compared against any predictions made. *Supervised learning* is a class of learning algorithms where ground truth labels are available and used to guide the learning process.

Consider input and output spaces X and Y respectively, the relationship between inputs and outputs can be expressed as the function mapping $f : X \to Y$. However, it is impossible to explicitly define f as the elements of X and Y are sampled often from real life observations. Supervised learning models aim to approximate f through a hypothesis function $h : X \to Y$ such that predictions made with the hypothesis function is close to the provided labels. Mathematically, this can be expressed as the minimization problem

$$h^* = \arg\min_h \sum_i L(h(x_i), y_i) \tag{3.1}$$

where $L : Y \times Y \to \mathbb{R}^{\geq 0}$ is a loss function defined over elements of Y. The hypothesis function h_θ takes on a parametric representation controlled by parameters $\theta \in H$, where H is the parameter space. Learning is achieved by finding the values of θ in the parameter space H such that $h_\theta(x_i)$ is close to y_i for all provided data. Although the mathematical formulation is identical to a constrained optimization problem, it should not be treated as one since the objective of any learning based algorithm is to find a generalized relationship, as opposed to a perfect fit, between the input and output spaces. Therefore it is common practice to separate a dataset into training, validation, and test sets. *Overfitting* occurs when a model is able to capture the behavior of the training set, but fails to reliably fit any additional data presented. This worsens when a dataset is noisy, as the model will treat noise as real data. *Underfitting* reflects the opposing scenario where a model fails to capture the underlying structure of the given data. Hence the goal of a

learning algorithm is *NOT* finding the minimizer of the defined loss function, but to use it as a guide towards generalization. These three scenarios are depicted in Figure (3.1).

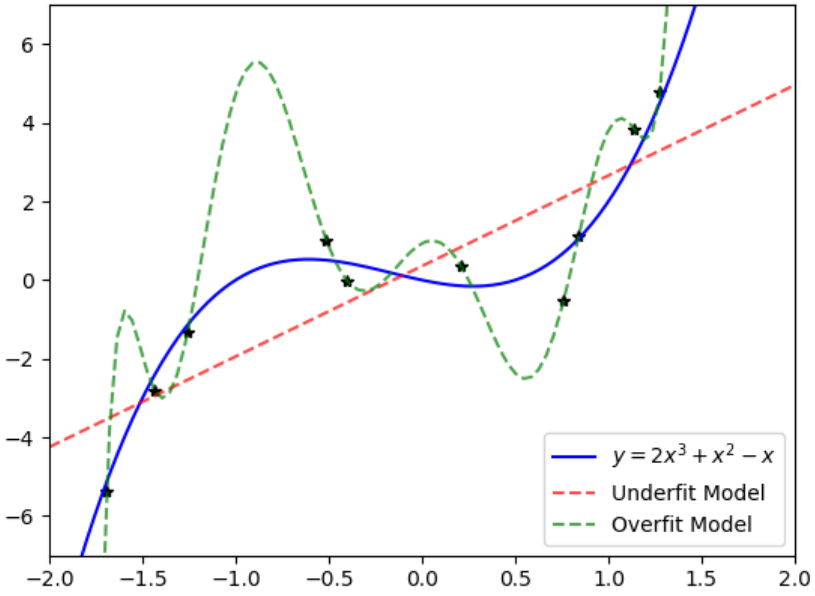

Figure 3.1: Fitting 10 data points to curve. Linear regression is shown as the red curve which fails to capture the behavior of the data points (undefitting). Although the green curve passes through all data points, it fails to generalize as it is too specific to the given data. The blue curve provides the best generalization as it captures most of the points with minimal variances. Data points are generated using $y_i = 2x_i^3 + x_i^2 - x_i + \xi(x_i)$ where $\xi(x_i)$ is a Gaussian noise term with $\mu = 0$ and $\sigma = 0.5$.

The simplest example of supervised learning is linear regression, a type of *regression* problem, where the goal is to find optimal values of α and β for the hypothesis function $h(x) = \alpha x + \beta$ by minimizing the sum of squared residuals

$$h^* = \arg\min_h \sum_i (h(x_i) - y_i)^2 = \arg\min_{\alpha,\beta} \sum_i (\alpha + \beta x_i - y_i)^2. \tag{3.2}$$

The solution of the above problem is well known through the normal equation

$$\theta = (A^T A)^{-1} A^T y \tag{3.3}$$

where $\theta = (\alpha, \beta)^T$, $y = (y_1, \ldots, y_n)^T$, and A is a matrix whose first column consists of all ones, and the second column contains all of x_i. However, hypothesis functions used in machine learning, and especially deep learning, do not have the luxury where the minimizer can be determined algebraically since the hypothesis function contains up to millions of parameters. Therefore, iterative algorithms such as gradient descent is employed to search for optimal parameter values. The second type of problem that supervised learning is used in are *classification* tasks. In classification problems, values in the output space Y are discrete valued as opposed to regression problems where the output space is typically over a continuous interval. For instance, consider the binary classification problem where the goal is to predict if a car exists in an image. The input space X is the set of images, and the output space is $Y = \{0, 1\}$ where 1 refers to the existence of a car, and 0 otherwise. The hypothesis function produces an output $y^* \in [0, 1]$ which can be interpreted as the probability of finding a car in a provided image. As such, it is natural to formulate this problem using conditional probability, by estimating the probability of the existence of a car given the input image, i.e.

$$\Pr(y_i|x_i) = h(x_i)^{y_i}(1 - h(x_i))^{1-y_i}. \tag{3.4}$$

Assuming that X and Y are independent and identically distributed (iid) random variables, the above can be generalized as

$$\Pr(Y|X) = \prod_{i=1}^{n} h(x_i)^{y_i}(1 - h(x_i))^{1-y_i}. \tag{3.5}$$

Since logarithms are strictly increasing functions, a loss function can be defined by taking the negative log of the above

$$\mathcal{L}(X) = -\sum_{i=1}^{n} y_i \log h(x_i) + (1 - y_i) \log(1 - h(x_i)). \tag{3.6}$$

This formulation is called the *maximum likelihood estimation* which is frequently employed in binary classification algorithms. The *softmax* function generalizes this approach

to multi-class classifiers by predicting the probability of each outcome given the provided input.

By including labels in the learning process, supervised learning paints a clear picture of what the learning outcome is. However, this is also a drawback as it requires labels to be pre-determined in the dataset. Furthermore, the assigned labels must reflect the goal of the formulated problem. In most cases, label assignment is a manual task which is extremely time consuming. This becomes even more problematic as the size of datasets are in the thousands, if not millions, as it requires large amounts of human labor.

Unsupervised Learning Learning algorithms where labels are not explicitly provided are called *unsupervised learning*. Since there are no labels to provide feedback, the learning algorithm's goal is to discover some form of structure based on only the input. In essence, the input space X can be viewed as a set of data points sampled from some data distribution p_{data}, and the learning algorithm's job is to either approximate this distribution, or uncover some unknown property of this distribution. Since there is no clear objective during the learning process, the outcome depends largely on the associated loss function. The two most common uses of unsupervised learning are *clustering* and *dimensionality reduction*.

Perhaps the simplest of clustering algorithms, k-means is a clustering algorithm that separates a set of data points into k clusters. The algorithm begins by randomly selecting k initial points as "means". Each data point is then assigned to the mean it is closest to based on a predefined metric. Once all data points are assigned, the centers of mass for all clusters are computed to be the new means. This procedure is repeated until convergence, which occurs when there are no changes in successive iterations.

Unsupervised learning also play a large role in *dimensionality reduction* tasks. Since complex data are very high dimensional in nature, finding an accurate low dimension representation of the data allows data to be stored and processed more efficiently. A

classical example is *principle component analysis* (PCA), where observations (potentially correlated) is stored in some matrix $A \in \mathbb{R}^{n \times m}$, where n is the number of observations and m is the number of features in each observation. The principle components are identified by choosing the k ($k < m$) most dominant right-singular vectors of A, or equivalently, the k most dominant eigenvectors of the covariance matrix $A^T A$. The m-dimensional data is then projected onto the subspace spanned by these vectors, thus reducing the dimensionality from m to k. Another example of unsupervised learning algorithms are *autoencoders*, an artificial neural network that learns a low-dimensional representation of the given data (encoding) then attempts to recover the original data (decoding). This *encoder* can be viewed as learning a mapping $\phi : X \to Z$ where $\dim(Z) < \dim(X)$. Since, by the pigeonhole principle, ϕ is not injective, therefore ϕ^{-1} does not exist and can only be approximated. Thus the *decoder* tries to learn the inverse mapping $\psi : Z \to X$ which will serve two purposes: 1. To ensure the representations by the encoder are meaningful, and 2. To learn a mapping that approximates ϕ^{-1}. The autoencoders are trained by minimizing the so-called *reconstruction loss*, defined as

$$\psi^*, \phi^* = \arg\min_{\psi,\phi} \mathop{\mathbb{E}}_{x \in X} [d(x, (\psi \circ \phi)(x))] \tag{3.7}$$

where $d : X \times X \to \mathbb{R}$ and $\mathbb{E}$ are the distance function and expected value over the set X. In most cases, the ℓ_2 norm is used as the distance measure. However, this can be generalized to any distance function.

3.2 Artificial Neural Networks

From the previous section, we described machine learning as finding optimal parameter values for the hypothesis function h_θ. Hypothesis functions are most commonly parameterized as artificial neural networks (ANN). In this section, we will explore the basics of ANNs, which are currently the most popular machine/deep learning models. We will discuss the two general classes of ANNs: Multi-layer perceptron (MLP), and convolutional

neural networks (CNN).

3.2.1 Multi-Layer Perceptron

Before talking about Multi-layer Perceptron, we must first start with the fundamental building blocks of an artificial neural network. A neural network comprises of a large network of interconnected computing units called neurons. These artificial neurons are inspired by biological neurons by mimicking their ability to transfer information. Each neuron receives a collection of inputs that are individually weighted. A weighted sum is computed within the neuron then passed to a nonlinear function σ called an activation function. When this is applied to a stack of neurons, called a *network layer*, this is effectively performing an affine transformation on the layer inputs, then passing the result to a nonlinear function to be evaluated element wise. Common examples of activations include the logistic (sigmoid) function $(1 + \exp(-kx))^{-1}$, hyperbolic tangent, rectified linear unit (ReLU), and LeakyReLU (Figure (3.2)). Figure (3.3) shows the schematic of a single artifical neuron.

For visualization purposes, multiple neurons are stacked vertically to form a network layer. The layers are then stacked horizontally to form the neural network itself. Figure (3.4) is an example of a four layer fully connected neural network. Note that in this neural network, information flows from the input layer towards the output layer in a single direction. In other words, the neural network does not feed the outputs of the model back into itself. Neural networks of this type are referred to as *feedforward neural networks*, and are represented as *directed acyclic graphs* (DAGs). Other names such as *fully-connected network* (FCN) and *multi-layer perceptron* (MLP) are often used interchangeably in literature.

During early research in deep learning, MLPs were shown to be very powerful as they were able to capture any desired representation provided that the network architecture was "deep enough". However, scalability became an issue since the number of weights

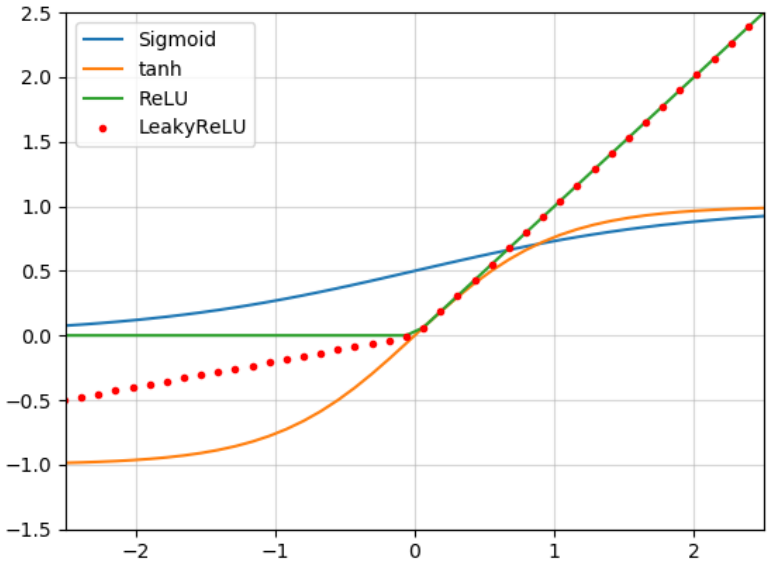

Figure 3.2: Examples of activation functions, which serve as gates to determined if information is allowed to be passed through a neuron.

scaled quadratically with the number of neurons. As a result, MLP should not be used in applications where high dimensional data is involved such as in signal and image processing. In the next section, we will explore a variant of neural networks, the *convolution neural network* which transforms input data through a series of discrete convolutions instead of affine transformation throughout immediate layers.

3.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) form a class of artificial neural networks that specialize in signal and image related tasks which require preservation of local spatial structure. Although MLP is more powerful, its fully-connected nature tends to establish relationships where relationships may not exist, such as opposite corners of an image. As a result, many connections in a MLP become redundant, thus unnecessarily increasing both computational requirements and training time.

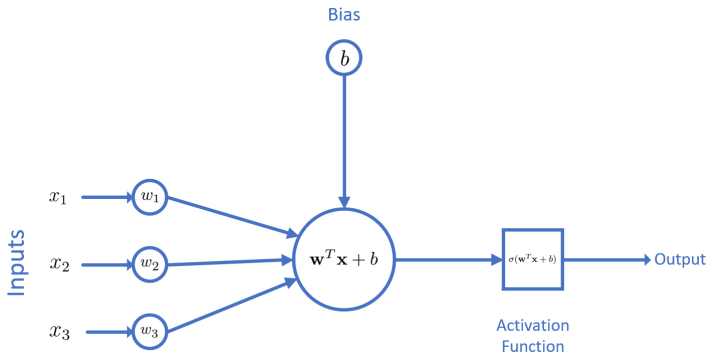

Figure 3.3: Schematic of a single artificial neuron with inputs $\mathbf{x} = [x_1, x_2, x_3]^T$ with corresponding weights $\mathbf{w} = [w_1, w_2, w_3]^T$ and the bias term $b \in \mathbb{R}$. This is followed by the activation function σ. The entire neuron can be summarized as the function $n(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$.

CNN attains its result by taking advantage of the discrete convolution operator

$$(f * g)[n] = \sum_{-\infty}^{\infty} f[m]g[n-m]. \tag{3.8}$$

In practice, the lower and upper bounds of the summation are finite because signals and images begin and terminate. For most applications, convolution is performed using a convolution kernel (or filter) against a signal or image. Using the convolution kernel as a weight matrix, a weighted sum over the neighboring regions of a fixed location is computed. This process is repeated across all locations to create a new signal/image. Explicitly, consider a 2D image I and kernel K, the discrete convolution produces an image S where the pixel intensity at pixel (i, j) is given by

$$S_{i,j} = (I * K)_{i,j} = \sum_m \sum_n I(m, n)K(i - m, j - n). \tag{3.9}$$

Convolutional Layers A convolutional layer consists of multiple kernels with unknown weights, where the number of kernels is often referred as the number of *output channels*. The learning process is to find the ideal weights for each convolution kernel. The output of each convolutional layer is called a *feature map*, which is optionally passed into an activation function then to the next layer similar to MLP. In most deep learning libraries (e.g. TensorFlow, PyTorch, Theano, etc.), the convolution operation is

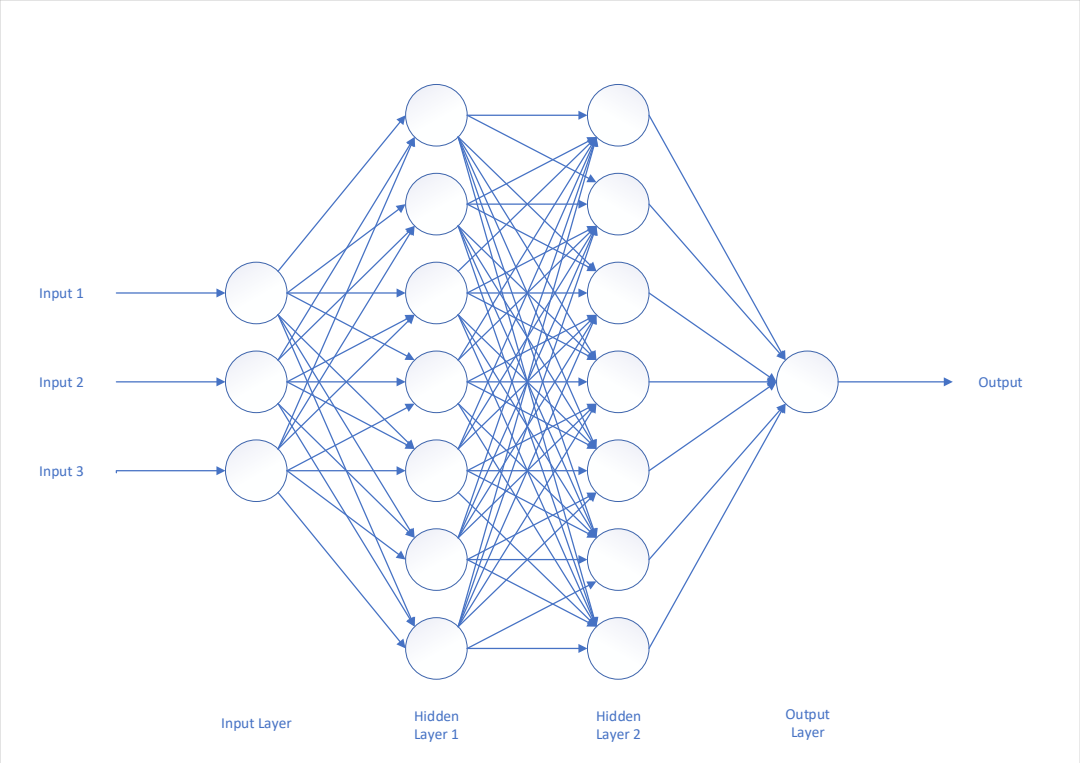

Figure 3.4: Example of a neural network consisting of an input layer, two hidden layers, and an output layer. The weights at each layer are represented by a single matrix of size $p \times q$, where p is the number of neurons in the previous layer and q is the number of neurons in the current layer. This neural network is equivalent to the expression $F(\mathbf{x}) = \sigma(\mathbf{W}_3^T \sigma(\mathbf{W}_2^T \sigma(\mathbf{W}_1^T \mathbf{x})))$, where the elements of $\mathbf{W}_i^T$ are weights assigned to each connection between adjacent layers.

implemented as cross-correlation instead, which only differs by a single sign:

$$(I \star K)_{i,j} = \sum_m \sum_n I(m,n) K(i+m, j+n). \tag{3.10}$$

Convolution with a fixed kernel is equivalent to computing the cross-correlation against the same kernel but in the reverse direction that the kernel traverses, effectively flipping the kernel. Since the weights of the kernels are learned, this flipped relationship is automatically discovered, making the two operations equivalent during implementation despite being mathematically different. In machine learning literature, both operations are referred to as convolution. This convention will be adopted for the remainder of this

thesis.

A distinct advantage of convolutional layer over fully-connected layer is the reduced number of learnable parameters. Consider a fully connected network with 128×128 grayscale image as its input and the first hidden layer consisting of 50 neurons. Each pixel of the image is treated as a node, and each of these nodes are connected to all 50 neurons in the first layer, resulting in $128 \times 128 \times 50 = 819,200$ learnable parameters at the first layer alone. The number of parameters continue to scale quickly as the number of intermediate layers increase. This steep computational requirement makes deep fully connected networks intractable. Convolutional layers prevent this through a technique called *weight sharing*, as the same convolutional kernel is applied to entire images. For instance, suppose that a convolutional layer contains 64 kernels of size 7×7. The number of learnable parameters is $7 \times 7 \times 64 = 3,136$ which is about 250 times less than the fully connected example. Furthermore, the use of convolutional layers make the number of learnable parameters independent of image size. Therefore, CNNs are able to accept inputs of varying sizes without the need to redesign the network model.

In order to preserve image dimensions, *padding* is required at the boundary. For most cases, the convolutional kernel have finite support therefore most of the zeros of the kernel can be discarded during implementation. This leads to kernel sizes that are significantly smaller than the image size. As a result, any time padding is required, the image will only need to be extended by a few pixels. Examples of different padding methods are shown in Figure (3.5).

From the definition of the discrete convolution, the kernels traverse through the entire image in steps of one pixel at a time. This step size is known as *stride*. By choosing stride to be greater than one, the convolution operation now produces a result whose resolution is reduced by a factor of the chosen stride. This approach can be applied to successive convolutional layers, allowing features to be learned over multiple spatial scales. Thus establishing a hierarchy of features as the input passes through the layers of the network.

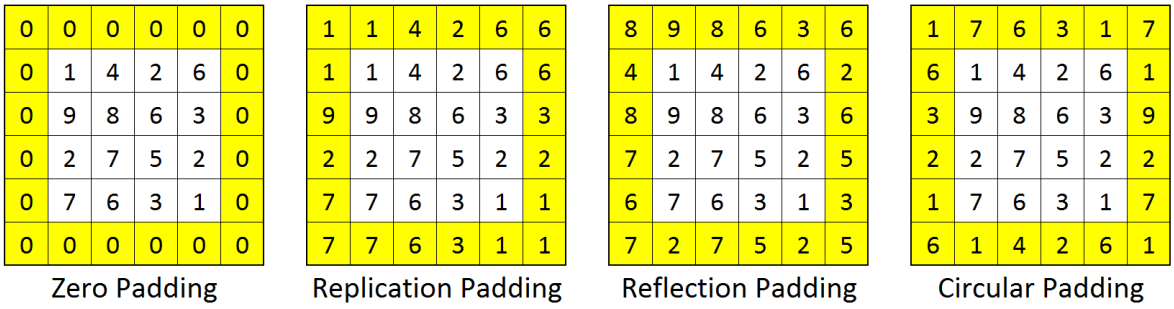

Figure 3.5: Examples of different padding schemes with corresponding boundary conditions (BC). The yellow blocks denote padded regions, and the white blocks denote the original data. From left to right: Zero-padding (Dirichlet BC), replication padding (Neumann BC), reflection padding (reflecting BC), circular padding (periodic BC).

Pooling Layers Pooling layers are intermediate network layers placed between convolutional layers. They are used to downsample activation maps to reduce dimensionality. They also, indirectly, help learned representations become invariant under small translations. The two main types of pooling methods are *average pooling* and *max pooling*. Both methods divide an image into small image patches, where the average and maximum values of the patch are calculated for average and max pooling respectively (Figure (3.6)).

Figure 3.6: Maxpooling and average pooling over a small image patch.

3.3 Training a Neural Network

3.3.1 Optimization

The deep learning framework can be summarized as finding optimal parameter values θ such that the hypothesis function h_θ minimizes some pre-determined loss function $\ell(\theta)$. Thus, optimization algorithms are essential tools of the learning process.

Gradient-based methods are optimization algorithms that rely on derivatives, where the degree of the method rests on the highest degree of the derivative required for the algorithm. For a basic overview of gradient-based methods, we refer to Section (2.2.2). In deep learning, gradient descent form the basis of most optimization algorithms. Recall that gradient descent updates the learning parameters through the recursive step

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}(\theta^{(k)}) \tag{3.11}$$

where γ is the learning rate, and $\mathcal{L}$ is a pre-defined loss function. Since the primary goal of machine learning is to discover some underlying pattern of data, the model must be trained using the entire dataset in order for the uncovered pattern to be meaningful. However, sizes of modern datasets range from Megabytes to Terabytes, making it next to impossible to store entire datasets in memory for processing especially for high dimensional data such as images. *Stochastic gradient descent* (SGD) bypasses this problem by randomly dividing the dataset into *minibatches*, where parameter updates are performed using minibatches as opposed to the entire dataset at once. From a statistical perspective, the dataset is viewed as some unknown probability distribution p_{data}, and minibatches are *independent and identically distributed* (i.i.d.) random variables sampled from this distribution. Gradients computed with minibatches are used to approximate of the true gradient with p_{data}. In other words, the update step becomes

$$\theta^{(k+1)} = \theta^{(k)} - \gamma \nabla \mathcal{L}^*(\theta^{(k)}) \tag{3.12}$$

where $\nabla \mathcal{L}^*(\theta^{(k)}) \approx \nabla \mathcal{L}(\theta^{(k)})$ is the minibatch approximation of the gradient. SGD, how-

ever, suffers from the same shortfalls as the standard gradient descent such as exploding and vanishing gradient. Exploding gradient occurs when the gradient approximations have large errors, causing large subsequent updates to network parameters leading to divergence. On the other hand, vanishing gradient occurs when gradients are very close to being the zero vector. Therefore, the update step makes very small changes to the previous iteration, leading to a very slow training process. Many improvements to SGD have been made to combat these two phenomena. Popular techniques such as Adagrad [32], Momentum [33], and Adaptive Moment Estimation (Adam) [34] have shown great success and have become the current standard for deep learning optimization. A comprehensive list of such methods can be found in the paper by Ruder [35].

3.3.2 Backpropagation

Traditionally, gradient descent requires gradients to be computed explicitly. In deep learning, this becomes an impossible task due to the sheer volume of parameters that must be tuned. *Back-propagation*, or simply *backprop*, is an algorithm that numerically computes gradients by taking advantage of chain rule from elementary calculus.

To demonstrate how back-propagation works, we first need to understand how information flows when a model is being trained. Training a network is an iterative process, where each iteration comprises of two stages: forward and backward pass. In the forward pass, inputs are provided to the network, where operations in each network layer are evaluated sequentially. In most cases, each neuron represents a differentiable function, and local Jacobians (with respect to the neuron's input) are computed and stored within the neuron. This process continues until the final layer is reached, where the numerical value of the loss is computed based on the prescribed input. In the backward pass, information flows in the reverse order starting with the computed loss. The gradient of the loss with respect to the input of each node are calculated by multiplying local Jacobian (computed in the forward pass) and the gradient of the loss with respect to the output of the node.

The process is repeated until all gradients are computed. Figure (3.7) demonstrates this process over a single neuron.

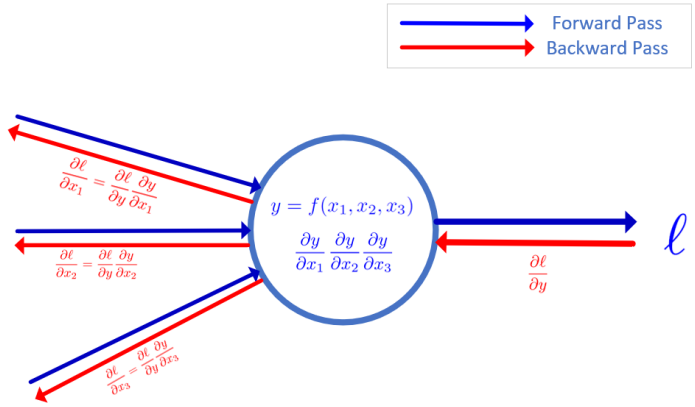

Figure 3.7: Backpropagation mechanism for a single neuron. In the forward pass, the output of the network $y = f(x_1, x_2, x_3)$ is computed, where $f : \mathbb{R}^n \to \mathbb{R}$ denotes the function mapping of the neron. During this step, the partial derivatives $\frac{\partial f}{\partial x_i}$ are computed. In the backward pass, $\frac{\partial \ell}{\partial y}$ is computed. The partial derivatives of the nodes in the preceeding layers are then computed via the chain rule $\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial y}\frac{\partial y}{\partial x_i}$.

In practice, users are only required to design the model used in the forward pass since gradients and Jacobian matrices are implicitly calculated by the chosen machine learning library. For functions that are not differentiable (such as the absolute value function), a subgradient value is typically assigned at points of non-differentiability.

Chapter 4

Discontinuity-Preserving Image Registration

4.1 Motivation

In Chapter 2, we explored deformable image registration techniques where uniform smoothness constraints are applied to the displacement field u. Similarly, parametric methods such as affine and thin plate spline imposes similar types of restrictions. A common behavior is that these restrictions are applied to the entire image domain which is not appropriate in many scenarios. For instance, consider two chest x-ray images of the same patient taken at separate times. Naturally, one would expect noticeable lung movement simply due to respiration. At the same time, we do not expect large movements in locations when rigid bone structures exist such as the spine. Registration techniques that impose global smoothness constraints assume that everything inside the image domain move in similar fashions. This leads to inaccurate registration results which may have significant impact on clinical decisions, especially during image guided surgery. Ideally, we must consider local image regions where movement discontinuities are expected. The modelling of motion discontinuities have had an increase in attention over recent years,

with the primary focus based on movement exhibited by organs with sliding interfaces [36, 37, 38, 39]. In this chapter, we explore discontinuity-preserving image registration by examining how the displacement field should behave near regions of discontinuities, then develop a model that preserves these desired behaviors.

4.2 Model

In our model, we use the traditional image framework as a starting point. Recall that the image registration problem is framed as the minimization problem

$$\arg\min_{u\in L^2(\Omega)} \mathcal{J}[u] = \arg\min_{u\in L^2(\Omega)} \left[\mathcal{D}(\mathcal{T}(x+u(x)), \mathcal{R}(x)) + \alpha\mathcal{S}[u]\right]. \tag{4.1}$$

The primary focus is to develop a regularizer capable of preserving discontinuous displacements while also maintaining local smoothness. This regularizer will be used as one of the terms of the loss function $\mathcal{L}$ which will be defined in Section (4.2.2). We determine the requirements of the regularizer by simply relating it to physical scenarios. Generally speaking, these scenarios can be separated into three distinct cases: Tissue/Organ interior, sliding organs, and tissue movement against solid structure. Although it is tempting to define a regularization term for each of the three cases, doing so leads to regularization terms competing against one another which makes hyper-parameter tuning almost impossible. Therefore, the ideal situation is to define a regularizer that encompasses all three cases simultaneously.

4.2.1 Framework and Network Architecture

Our model follows a framework popularized by Voxelmorph [40]. Let I_F and I_M denote fixed and moving images. We find a function $g_\theta(I_F, I_M)$ that produces the displacement field $\mathbf{u}$, i.e. $\mathbf{u} = g_\theta(I_F, I_M)$. The deformation ϕ can then be expressed as the mapping $\phi = Id + \mathbf{u}$ where Id is the identity mapping. The deformation field is applied to

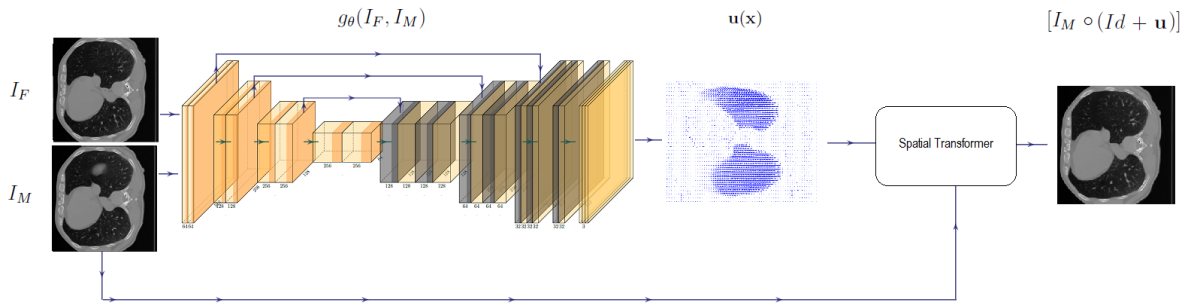

Figure 4.1: Overview of the model. Fixed and moving images I_F, I_M are passed into a convolutional neural network g_θ which produces the displacement field $\mathbf{u}(\mathbf{x})$. The spatial transformer morphs I_M based on the displacement field. The loss is measured over the dissimilarity between the fixed and morphed moving images, as well as additional penalty functions defined over $\mathbf{u}$.

I_M to produce the warped image $I_M \circ \phi$ where $I_F(\mathbf{x})$ is similar to $[I_M \circ \phi](\mathbf{x})$ for all voxel locations $\mathbf{x} \in \Omega$. Since ϕ may map the original coordinate system to non-integer valued voxel locations, interpolation is required to warp I_M under ϕ. This step is more frequently known as *spatial transform* in deep learning literature. In our experiments, we use trilinear interpolation due to its simplicity. An overview of the model is shown in Figure (4.1).

The function g_θ is modeled using a convolutional neural network where θ denotes the network parameters. The neural network follows a modified version of U-Net [41], which contains an encoder and a decoder structure that mirror each other and are connected by skip connections at each layer (Figure (4.2)). The encoder/decoder architecture is motivated by image pyramid techniques in traditional image registration, as each encoding and decoding layer operates from coarse to fine representations of the input.

The encoder consists of three convolution layers by applying $3 \times 3 \times 3$ convolutions with stride 2 for downsampling, followed by LeakyReLU with slope of 0.2 at each layer in order to prevent vanishing gradients for negative layer outputs experienced with Rectified Linear Units [42]. Each convolution layer has 32 output channels except the first layer which contains 16 output channels.

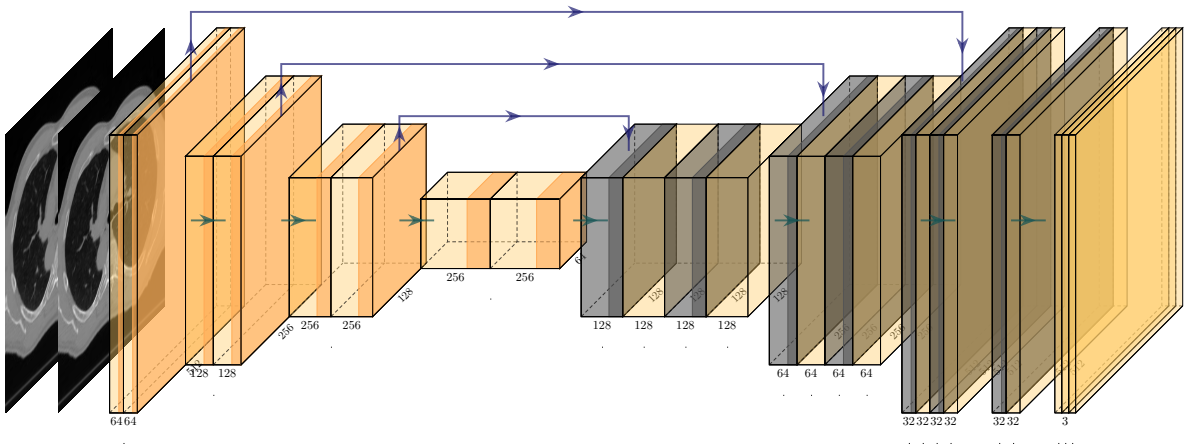

Figure 4.2: Network architecture of g_θ based on a modified version of U-Net. The network receives I_M and I_F to produce the displacement field $\mathbf{u}$. The input and output of the network are of dimensions $D \times H \times W \times 2$ and $D \times H \times W \times 3$ respectively. The architecture consists of a contractive path (encoder) and a mirroring expansive path (decoder) connected by skip connections at each layer.

The decoder follows a similar structure as the encoder but in reverse order. In the first decoding layer, we simply use the output of the final encoding layer as the input. In subsequent decoding layers, we first upsample the output of the previous decoding layer. So-called *skip connections* act as bridges that connect mirroring encoding and decoding layers together. They are constructed by concatenating outputs of an encoding layer with the input of the opposite decoding layer. This effectively uses representations of the encoding layers to encourage more precise outputs in the decoding layers. Similar to the encoder, each decoding layer applies $3 \times 3 \times 3$ convolutions followed by LeakyReLU of slope 0.2, but with stride 1 to preserve resolution at each layer. The output of the final decoding layer is passed into an additional convolution layer with 3 output channels, where each output channel contains the coordinate components of the displacement field $\mathbf{u}$.

4.2.2 Loss Function

We train our model using a loss function in the form

$$\mathcal{L}(I_F, I_M, \mathbf{u}) = \lambda_{\text{sim}}\mathcal{L}_{\text{sim}}(I_F, I_M, \mathbf{u}) + \lambda_{\text{disc}}\mathcal{L}_{\text{disc}}(\mathbf{u}) + \lambda_{\text{mag}}\mathcal{L}_{\text{mag}}(\mathbf{u}), \tag{4.2}$$

where $\mathcal{L}_{\text{sim}}$ measures image dissimilarity, $\mathcal{L}_{\text{disc}}$ is a discontinuity preserving regularizer, and $\mathcal{L}_{\text{mag}}$ is a second loss term that manages the (ir)regularities in the magnitude of the displacement fields. $\lambda_{\text{sim}}, \lambda_{\text{disc}}$, and λ_{mag} are corresponding regularization constants, or *hyper-parameters*. Similar to the terms $\mathcal{D}$ and $\mathcal{S}$ discussed in Chapter 2, the behavior of the transformation model depends heavily on the choices of $\mathcal{L}_{\text{sim}}$ and $\mathcal{L}_{\text{disc}}$.

Local Normalized Cross Correlation The correct choice of $\mathcal{L}_{\text{sim}}$ is of utmost importance. We avoided norm-based distance measures, such as MAE, as they originate from direct intensity comparisons between fixed pixels. On the other hand, NCC accounts for translations on a global scale, which diminishes the effect of local irregularities. We find a middle ground through local normalized cross correlation (LNCC) [40]. In LNCC, we first fix a single pixel in the image domain. Normalized cross correlation is calculated with a small neighborhood around the pixel. This step is repeated for all pixels in the image domain then summed together, effectively comparing local image structure rather than direct pixel comparison. Mathematically, LNCC is defined as

$$\text{LNCC}(I_M, I_F) = \sum_{x\in\Omega} \frac{\left[\sum_{y\in\mathcal{N}(x)} \left(I_M(y) - \mu_M(x)\right)\left(I_F(y) - \mu_F(x)\right)\right]^2}{\left[\sum_y \left(I_M(y) - \mu_M(x)\right)^2\right]\left[\sum_y \left(I_F(y) - \mu_F(x)\right)^2\right]} \tag{4.3}$$

where x is any voxel in the image domain Ω, $y \in \mathcal{N}(x)$ are the neighborhood points around voxel x, and $\mu_M(x)$ and $\mu_F(x)$ are the average intensities of the region. If the first summation of Equation (4.3) is removed, this simply refers to the normalized cross correlation of the image patch centered around x_i. Note that LNCC is maximized when $I_F = I_M$ which measures similarity, thus we define the dissimilarity measure as $\mathcal{L}_{\text{sim}} = 1 - \text{LNCC}$.

Discontinuity-Preserving Regularization Since the distance measure $\mathcal{L}_{\text{sim}}$ only quantifies disagreement between two images, any discontinuity-preserving properties must be encoded in the regularizer. When designing the regularizer, we first assume that there are no topological changes, i.e. nothing is introduced nor destroyed. We then consider the requirements based on the three physical scenarios: 1. Homogeneous movement, 2. Movement along rigid structures, and 3. Sliding organs.

The three physical scenarios help us define the requirements for our regularizer. Firstly, the regularizer must preserve smooth local deformations. This occurs when a local region is comprised of a single organ or tissue. As this represents a single physical object, deformation must be locally smooth in order to be physically meaningful. Secondly, the regularizer must not penalize large local changes in deformation magnitude as long as movement is in a similar direction. This is to mimic the movement of soft tissues or organs against rigid structures such as the rib cage or the spinal column. Finally, the regularizer must be able to account for movements in the opposite directions along organ boundaries. This final requirement is perhaps the most significant as there are many scenarios where sliding organs exist. Common examples include the sliding of the lungs against the chest wall during the respiratory cycle, and the movement of organs against one another in the abdominal region. Figure (4.3) summarizes the feasibility and unfeasibility conditions visually.

Fortunately, the three conditions can be accomplished by associating pairs of deformation vectors to a simple geometric object: the parallelogram. More specifically, consider two deformation vectors $u_i = \mathbf{u}(x_i)$ and $u_j = \mathbf{u}(x_j)$. The area of the parallelogram spanned by u_i and u_j is maximized when u_i and u_j is orthogonal to one another, and minimized when they are parallel. This immediately satisfies the first condition of local homogeneity. The area is minimized whenever u_i and u_j are parallel. This occurs whenever the two vectors point in the same or opposite directions regardless of their respectively magnitude, which is very desirable as it simultaneously satisfies the second

(a) (b) (c)

Figure 4.3: Desired behaviors of the discontinuous displacement field. Figure (4.3(a)) demonstrates local homogeneity which is expected within organs. Figure (4.3(b)) allows displacement vectors of different magnitudes as long as they are in a similar direction, which represents soft tissue moving against rigid structures. Figure (4.3(c)) depicts sliding boundary conditions as displacement vectors on opposite sides of the boundary travel in opposite directions.

and third requirements. Our regularizer $\mathcal{L}_{\text{disc}}$ takes the form

$$\mathcal{L}_{\text{disc}} = \sum_{i,j=1}^{N} g(\mathcal{P}(u_i, u_j)) \tag{4.4}$$

where $\mathcal{P}$ the unsigned area of the parallelogram spanned by u_i and u_j, and $g : \mathbb{R} \to \mathbb{R}$ is a strictly increasing function satisfying $g(0) = 0$. In two-dimensions, $\mathcal{P}$ can be computed as

$$\mathcal{P}^{(2)}(u_i, u_j) = |\det(u_i, u_j)|, \tag{4.5}$$

and in three-dimensions,

$$\mathcal{P}^{(3)}(u_i, u_j) = \|u_i \times u_j\|_2 \tag{4.6}$$

where $\times$ denotes the cross product and $\|\cdot\|_2$ denotes the Euclidean norm. We propose the regularizer

$$\mathcal{L}_{\text{disc}} = \sum_{i,j=1}^{N} \frac{1}{2} \log\left(1 + \mathcal{P}(u_i, u_j)^2\right) k(x_i, x_j) \tag{4.7}$$

where $k(x_i, x_j)$ is a decreasing weight function relative to the proximity between the locations x_i and x_j. For our experiments, we choose the C^4 Wendland kernel, defined as

$$k(x_i, x_j) = k(r) = (1 - r)_+^6(35r^2 + 18r + 3) \tag{4.8}$$

where $r = |x_i - x_j|$ and

$$r_+ = \max(0, r). \tag{4.9}$$

Although there are many candidates in choosing the weight function, the Wendland kernel is chosen for two main purpose: 1) Has compact support, which allows numerical implementations that do not require clipping small values to zero, and 2) Decays similarly to a Guassian function. For additional details and properties of the Wendland kernel, we refer to the manuscripts published by Wendland [27].

Restricting deformation magnitude During preliminary stages of our experiments, we noticed that deformations in large dark image regions (background of CT image, for instance) behave erratically. We found that imposing an additional magnitude-based regularizer is needed to suppress this unpredictable behavior. Thus we add the following term to our loss function

$$\mathcal{L}_{\text{mag}}(u) = \max_i(\|u_i\|_2). \tag{4.10}$$

This implicitly imposes an upper bound on the magnitude of u depending on the regularization parameter set during training. Evidently, this additional regularizer may become problematic for coarse registration where large-scale movement may occur. However, since this is aimed towards addressing local discontinuity, it is safe to assume that deformations remain relatively small.

4.2.3 Numerical Implementation

In this section, we discuss the implementation details of the proposed loss function

$$\mathcal{L} = \lambda_{\text{sim}}\mathcal{L}_{\text{sim}} + \lambda_{\text{disc}}\mathcal{L}_{\text{disc}} + \lambda_{\text{mag}}\mathcal{L}_{\text{mag}} \tag{4.11}$$

defined in equations (4.3), (4.7), and (4.10) respectively. Although the loss function is mathematically straight forward, the numerical implementations of $\mathcal{L}_{\text{sim}}$ and $\mathcal{L}_{\text{disc}}$ become much trickier in a deep learning environment due to memory and computational restrictions. Thus we re-formulate each term of the loss function that enables us to take advantage of the built-in parallelization of PyTorch. Although this does not decrease the total number of floating point operations required, a parallelized formulation reduced the computational requirements dramatically (from approximately 100 to around 9 Gigabytes of GPU memory), while training speed increased by about 70 times.

Local Normalized Cross Correlation Recall that LNCC is given as

$$\text{LNCC}(I_M, I_F) = \sum_{x\in\Omega} \frac{\left[\sum_{y\in\mathcal{N}(x)} \left(I_M(y) - \mu_M(x)\right)\left(I_F(y) - \mu_F(x)\right)\right]^2}{\left[\sum_{y\in\mathcal{N}(x)} \left(I_M(y) - \mu_M(x)\right)^2\right]\left[\sum_{y\in\mathcal{N}(x)} \left(I_F(y) - \mu_F(x)\right)^2\right]} \tag{4.12}$$

where y are the voxels in a n^3 window centered around x. We first ignore the summation in the front, which effectively forces our focus on the cross correlation of a single image patch centered around some voxel x. Expanding the numerator, we get

$$\sum_{y\in\mathcal{N}(x)} \left(I_M(y) - \mu_M(x)\right)\left(I_F(y) - \mu_F(x)\right) \tag{4.13}$$

$$= \sum_{y\in\mathcal{N}(x)} I_M(y)I_F(y) - \mu_M(x)\left(\sum_{y\in\mathcal{N}(x)} I_F(y)\right) - \mu_F(x)\left(\sum_{y\in\mathcal{N}(x)} I_M(y)\right) + n^3\mu_M(x)\mu_F(x).$$

The same is done for the two terms in the denominator, giving us

$$\sum_{y\in\mathcal{N}(x)} \left(I_M(y) - \mu_M(x)\right)^2 = \sum_{y\in\mathcal{N}(x)} \left(I_M(y)\right)^2 - 2\mu_M(x)\left(\sum_{y\in\mathcal{N}(x)} I_M(y)\right) + n^3\left(\mu_M(x)\right)^2,$$

$$\sum_{y\in\mathcal{N}(x)} \left(I_F(y) - \mu_F(x)\right)^2 = \sum_{y\in\mathcal{N}(x)} \left(I_F(y)\right)^2 - 2\mu_F(x)\left(\sum_{y\in\mathcal{N}(x)} I_F(y)\right) + n^3\left(\mu_F(x)\right)^2,$$

where

$$\mu_M(x) = \frac{1}{n^3}\sum_{y\in\mathcal{N}(x)} I_M(y), \tag{4.14}$$

$$\mu_F(x) = \frac{1}{n^3}\sum_{y\in\mathcal{N}(x)} I_F(y). \tag{4.15}$$

Summation	Convolution Operation
$\sum I_M(y)I_F(y)$	$(I_M \odot I_F) * K =: \mathbf{I}_{MF}$
$\sum I_M(y)$	$I_M * K =: \mathbf{I}_M$ (and similarly for I_F)
$\sum (I_M(y))^2$	$(I_M \odot I_M) * K =: \mathbf{I}_M^2$ (and similarly for I_F)
$\mu_M(x)$	$\frac{1}{n^3}(I_M * K) =: \boldsymbol{\mu}_M$ (and similarly for μ_F)

Table 4.1: Association between each summation term and their corresponding convolution operation. Here, $\odot$ denotes element-wise multiplication (Hadamard product), and $*$ is the discrete convolution.

Note that each summation is unweighted with respect to neighboring voxels. This allows us to compute an image of patch-wise cross correlation (I_{CC}) through a series of linear convolutions against a discrete kernel K, where K is a $n \times n \times n$ array of ones. Each summation is associated with a corresponding convolution operation and is summarized in Table (4.1). The image of patch-wise cross correlation is

$$I_{CC} = \frac{(\mathbf{I}_{MF} - \boldsymbol{\mu}_M \mathbf{I}_F - \boldsymbol{\mu}_F \mathbf{I}_M + n^3 \boldsymbol{\mu}_M \boldsymbol{\mu}_F)^2}{[\mathbf{I}_M^2 - 2\boldsymbol{\mu}_M \mathbf{I}_M + n^3(\boldsymbol{\mu}_M)^2]\,[\mathbf{I}_F^2 - 2\boldsymbol{\mu}_F \mathbf{I}_F + n^3(\boldsymbol{\mu}_F)^2]}. \tag{4.16}$$

It is important to note that all arithmetic in Equation (4.16) are element-wise operations, in order for I_{CC} to be well defined. The overall cross correlation (LNCC) is the mean over all the elements of I_{CC}

$$LNCC(I_M, I_F) = \frac{1}{h \times w \times d} \sum_{x \in \Omega} I_{CC}(x). \tag{4.17}$$

Algorithm (1) in Appendix (B) outlines the pseudocode of this implementation.

Discontinuous Loss We apply a similar approach in implementing the discontinuous loss. In a serial approach, the parallelograms are computed by comparing the vector at a fixed voxel against all its neighboring vectors. This is then repeated for all voxels of the image. Instead, we take a reverse approach, by comparing all voxels against a fixed neighbor (relative to voxel location). By fixing the neighbors, neighboring vectors can be

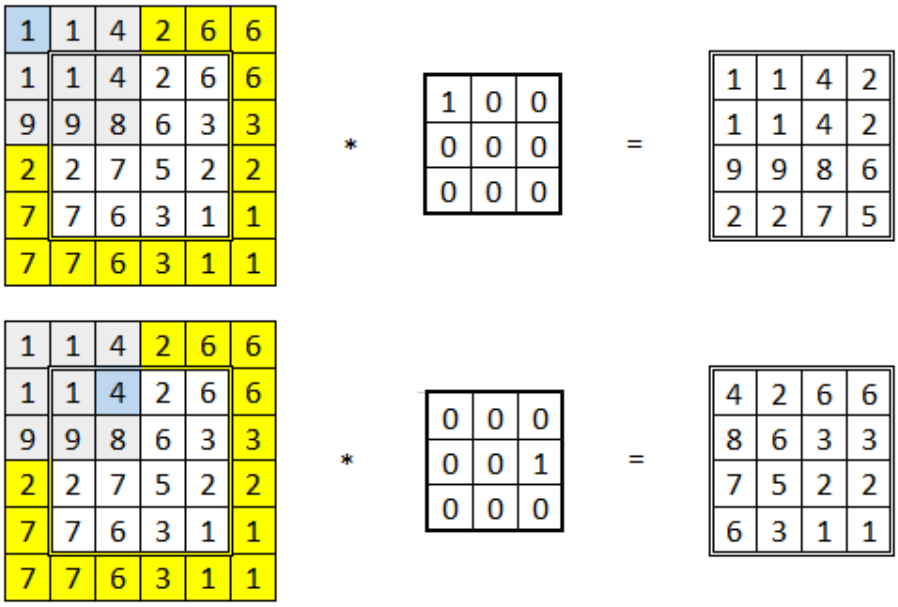

Figure 4.4: Extracting vectors from the top-left neighbor (top) and right neighbor (bottom) using convolution. From left to right: Original image with replication padding, convolution filter, result. Yellow indicates the padded regions of the input. Grey indicates the window that the filter presides in the initial convolution step, and light blue indicates the value extracted during that step.

extracted in parallel using convolution, where the vectors extracted are determined by the convolution filters. Figure (4.4) demonstrates this approach in 2D.

In order to extract all neighboring information, a filter must be created for each fixed neighbor, except the center (since this is simply the input). For example, consider a $10 \times 10 \times 10$ displacement field. This displacement field is numerically stored as a $10 \times 10 \times 10 \times 3$ tensor, where the last index contains the components of the displacement field along each coordinate direction. Suppose that we are interested in extracting displacement vectors in a $3 \times 3 \times 3$ neighborhood, we will then require $(3 \times 3 \times 3 - 1) = 26$ filters, with each filter corresponding to a fixed neighbor. Convolution is then applied to each of the three slices of the displacement field against the 26 filters, resulting in an output of size $26 \times 10 \times 10 \times 10 \times 3$. The parallelograms are computed by taking the norm of the

cross product between each $10 \times 10 \times 10 \times 3$ slice of the output against the input. The remaining operations are then applied element-wise, until all neighboring displacement vectors are accounted for. Algorithm (2) in Appendix (B) outlines the pseudocode of this implementation.

4.3 Experiments

4.3.1 Setup and Datasets

Our model is implemented using PyTorch 1.3.0 and trained using an Nvidia GeForce GTX 1080Ti with 11 Gigabytes of graphics memory. CPU tests are performed on a workstation with the processor Intel Xeon E5-1620 at 3.7 GHz. We trained our model using Adam optimizer [34] with $\lambda_{\text{sim}} = 100, \lambda_{\text{disc}} = \lambda_{\text{mag}} = 1$, and learning rate 10^{-4}.

The model is evaluated over 4DCT datasets provided by DIR-Lab [5, 6] and the POPI-model (Point-validated Pixel-based Breathing Thorax Model) [7]. The DIR-Lab Reference 4DCT datasets contain ten sets of image volumes of sizes 256×256 and 512×512 with various number of axial slices (average of 100 and 128 for the two respective resolutions) for each resolution level. To account for these variations, we only keep the middle 96 axial slices of the 256×256 volumes, and the middle 112 axial slices of the 512×512 volumes. Each set of image volumes are taken over 10 time steps over the period of a single respiratory cycle. Since the input of the model is a pair of image volumes, I_F is chosen as the image volume with a randomly chosen case number and time step, and I_M is selected based on the same case number with a different time step. By choosing eight cases as training data, this allows $8 \times 10 \times 9 = 720$ training samples and $2 \times 10 \times 9 = 180$ test samples, despite only having ten available cases.

The POPI-Model contains six image volumes of sizes 512×512 with 140 to 190 axial slices. For consistency, we only keep the middle 136 axial slices and use five of the six cases as training data. We follow the same approach as DIR-Lab in choosing I_F and I_M.

Since the image volumes of all three datasets are taken in quick succession throughout a single respiratory cycle, it can be assumed that pairs of chosen image volumes exhibit minimal large scale global movements.

4.3.2 Qualitative Results

We show a series of registration results using the proposed model. Since the image data are 3D image volumes, the results are presented by choosing axial, sagittal, and coronal slices that capture the majority of internal organ structure. Displacement magnitudes are computed via the Euclidean norm of the displacement vectors at each voxel location.

Figure (4.5) shows that our model performs registration very well qualitatively as the registered image is almost indistinguishable from the fixed image with a quick visual inspection. Upon taking a closer look, the model sometimes fails to accurately register regions that contain heavy location variation. We suspect the inconsistency is due to the model's inability to differentiate the difference between image intensity variations and image noise.

Additional results are included in Appendix (C).

4.3.3 Quantitative Results

The most common metrics of measuring the accuracy of image registration algorithms are the Jaccard index, Dice's coefficient, and the target registration error (TRE) [43]. The Jaccard index is computed by taking the ratio between the cardinalities of the union and intersection of two sets, formally this is defined as

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{4.18}$$

Dice's coefficient is also computed using cardinalities of sets, but with an alternative approach. It is defined as

$$\text{Dice}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{4.19}$$

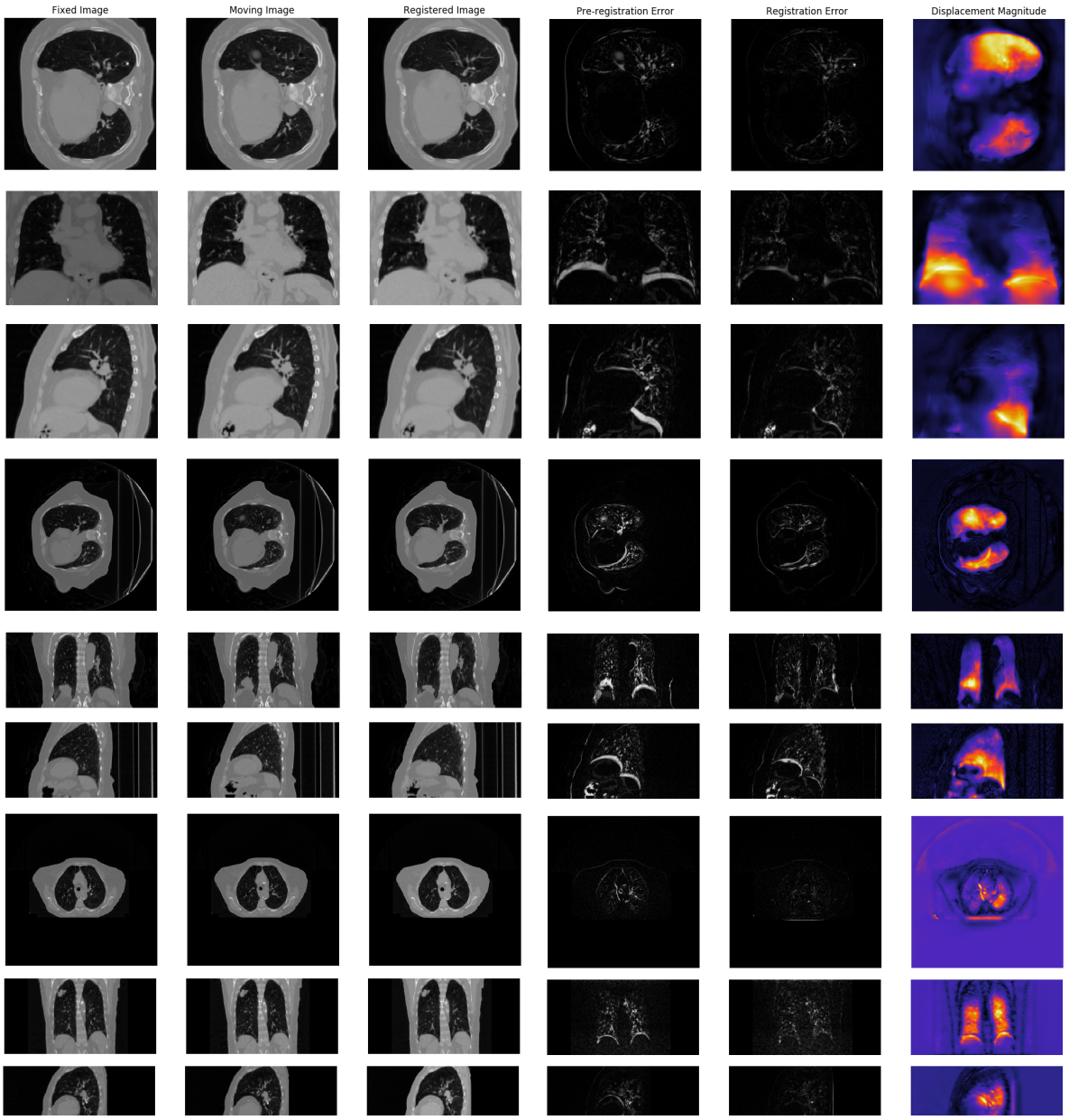

Figure 4.5: Qualitative results of proposed model. From left to right: fixed image I_F, moving image I_M, registered image $I_M \circ \phi$, absolute error before registration $|I_F - I_M|$, absolute error after registration $|I_F - I_M \circ \phi|$, heatmap of displacement field $\mathbf{u}$. Two images with no registration error corresponds to a pitch black image.

Both metrics are bounded between 0 and 1, where 1 represents perfect overlap and 0 represents no similarities. In the context of image registration, these sets are the voxel locations of regions of interest in the fixed image and registered image. These regions are typically selected via an image segmentation algorithm prior to registration. However, since the focus is not the registration of a particular organ, but rather discontinuities in the displacement field, the Jaccard index and Dice coefficient do not accurately capture the objectives of this model.

Instead, registration accuracy is quantified using the target registration error (TRE). In our datasets, landmarks were carefully placed by clinical experts to identify important features within an image volume. TRE is computed by measuring the physical distances of every corresponding landmark pairs between fixed and moving images.

We compare our results quantitatively on the DIR-Lab and POPI datasets to the following methods: Free-Form Deformations (FFD) [1], isotropic parametric Total Variation (pTV) [2], and Sparse Kernel Marchines (SKM) [3]. For comparison, we fixed frame 1 as the fixed image, and register all remaining frames to the reference.

	DIR-Lab 256					DIR-Lab 512					POPI				
Frame	FFD	pTV	SKM	Base	Ours	FFD	pTV	SKM	Base	Ours	FFD	pTV	SKM	Base	Ours
0	1.01	0.92	1.06	1.10	1.04	0.79	0.62	0.59	0.77	0.65	0.79	0.72	0.66	0.77	0.76
2	0.99	0.89	0.85	0.94	0.91	0.81	0.63	0.57	0.73	0.64	0.81	0.71	0.65	0.73	0.74
3	1.29	1.34	1.32	1.26	1.24	1.14	0.99	1.01	0.97	1.00	1.14	1.12	1.17	1.08	1.11
4	1.26	1.27	1.25	1.23	1.26	1.11	0.92	0.93	0.96	0.95	1.11	1.01	1.07	1.04	1.07
5	1.27	1.29	1.35	1.29	1.31	1.11	1.00	1.01	0.99	1.02	1.11	1.11	1.13	1.10	1.16
6	1.31	1.17	1.18	1.27	1.25	1.20	0.90	0.89	1.02	0.92	1.20	1.03	1.00	1.11	1.06
7	1.36	1.19	1.22	1.25	1.30	1.20	0.95	0.93	1.00	1.01	1.20	1.06	1.05	1.09	1.13
8	1.10	1.05	0.94	1.04	1.07	0.88	0.73	0.67	0.78	0.79	0.88	0.84	0.75	0.88	0.90
9	1.09	0.97	0.99	1.07	1.09	0.92	0.70	0.75	0.78	0.80	0.92	0.81	0.83	0.86	0.89

Table 4.2: Target Registration Error (TRE) in millimeters (mm) against FFD [1], pTV [2], and SKM [3] on the DIR-Lab and POPI 4DCT Model. Baseline model is the same configuration but trained with total variation loss in place of $\mathcal{L}_{\text{disc}}$.

Table (4.2) show that our model performs on par with current state-of-the-art methods. Without redefining the discontinuous loss, we believe that further improvements can be made by additional parameter tuning. Table (4.3) shows the mean and standard deviation of TRE for each of the datasets.

	FFD	pTV	SKM	Baseline	Ours
DIR-Lab 256	1.19 (0.92)	1.12 (0.83)	1.13 (0.77)	1.16 (0.92)	1.16 (0.98)
DIR-Lab 512	1.02 (0.74)	0.83 (0.61)	0.82 (0.61)	0.89 (0.65)	0.86 (0.59)
POPI	1.02 (0.63)	0.93 (0.62)	0.92 (0.77)	0.96 (0.57)	0.98 (0.61)

Table 4.3: Mean and standard deviation (in brackets) of target registration error (TRE) in millimeters (mm) against FFD [1], pTV [2], and SKM [3] on the DIR-Lab and POPI 4DCT Model. Baseline model is the same configuration but trained with total variation loss in place of $\mathcal{L}_{\text{disc}}$.

4.3.4 Discontinuity Preserving versus Non-Preserving Model

We compare our discontinuity-preserving model with one that assumes global smoothness. As a baseline, we trained a second model using the DIR-lab dataset with an identical configuration, with the exception where the discontinuous loss $\mathcal{L}_{\text{disc}}$ is replaced with a total variation loss $\mathcal{L}_{\text{TV}}$ defined as

$$\mathcal{L}_{\text{TV}} = \sum_i \|\nabla u_i\|_2 \tag{4.20}$$

where the summation is over all voxels indexed by i. Figures (4.6) and (4.7) show comparisons between the model trained using $\mathcal{L}_{\text{TV}}$ and $\mathcal{L}_{\text{disc}}$. To emphasize movements, we registered two image volumes where the moving and fixed images corresponding to the beginning and end of exhalation phase respectively, when the lungs are the most dilated and compressed.

Upon examining the lung/vertebrae interface, where motion discontinuities are expected, displacement vectors over the lungs show significant movement while adjacent

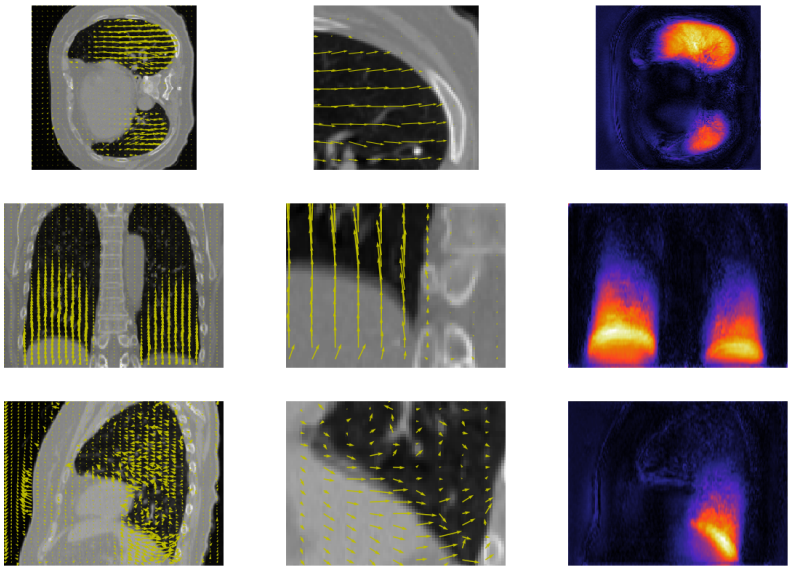

Figure 4.6: Qualitative results using $\mathcal{L}_{\text{TV}}$. Column 1 shows an overlay of the displacement field $\mathbf{u}$ over I_M. Column 2 shows a magnified region where transformation discontinuities are expected locally. Heatmaps of the displacement field's local magnitudes are shown in column 3.

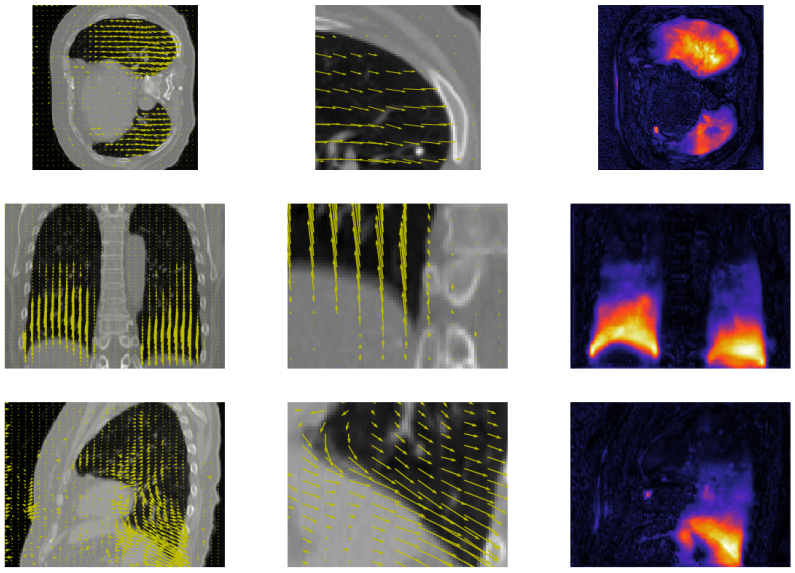

Figure 4.7: Qualitative results using $\mathcal{L}_{\text{disc}}$. Column 1 shows an overlay of the displacement field $\mathbf{u}$ over I_M. Column 2 shows a magnified region where transformation discontinuities are expected locally. Heatmaps of the displacement field's local magnitudes are shown in column 3.

vectors on the spine remain near motionless. A similar behavior was captured in the baseline model as well which was a surprising result. We suspect that this is caused by an imbalance between the orders of the numerical values between $\mathcal{L}_{\text{disc}}$ and $\mathcal{L}_{\text{TV}}$. This can be addressed by fine tuning the weights λ_{disc} and λ_{TV} individually. However, these are kept equal for the sake of a fair comparison between the two regularizers. A more noticeable difference can be seen when comparing the heatmaps generated by the two models, as the model trained with $\mathcal{L}_{\text{disc}}$ is visually sharper than the model trained with $\mathcal{L}_{\text{TV}}$. In particular, the transition exhibits a smoother gradient when transitioning from the inside to the outside of the lung in the model trained with $\mathcal{L}_{\text{TV}}$, suggesting that the discontinuous model does a better job at identifying the lungs' boundary.

4.3.5 Computation Time

Traditionally, image registration is a very time consuming process since registering image pairs require numerically minimizing a prescribed objective function. By employing a learning-based model, the required computation time is shifted almost entirely to the training step. Hence the registration process simply requires evaluating the trained model during inference, which is equivalent to performing a function evaluation.

With the proposed model, registration between image volumes of $256 \times 256 \times 96$ requires under a second to perform on a GPU, and requires approximately 16 seconds on a CPU. These times are significantly lower than the classical paradigms, which require upwards of two minutes on a GPU and an hour on a CPU. The time difference between the two approaches increases significantly with higher resolution image volumes, as the number of operations required scale at $O(z^d)$, where z is the resolution scaling factor and $d = 3$ is the number of dimensions in the image volume. We register two images and compare the computation time required using a our model (learning-based) and a classical model. The classical model is regularized using the diffusion regularizer, where the objective functional is numerically optimized over 1000 iterations. The mean registration

times across the three datasets are outlined in Table (4.4). We can immediately notice the significant reduction of registration time using a learning-based model, especially in a GPU-accelerated environment. Improvements on computation time can be made further by investigating whether the network architecture can be made smaller without jeopardizing registration accuracy.

	Learning-based		Inverse Model	
	GPU	CPU	GPU	CPU
DIR-lab 256	0.3306	15.6981	82.5739	5724.36
DIR-lab 512	1.3794	63.1433	532.4132	-
POPI Model	1.6722	76.4478	702.8592	-

Table 4.4: Comparison of registration time between learning-based model and inverse model measured in seconds. For the learning-based model, we used our proposed model for evaluation. For the inverse model, we perform pairwise registration with diffusion regularizer over 1,000 iterations. The inverse model is evaluated using the AIRLab framework [4]. CPU times for the classical model over DIR-Lab 512 and POPI Model are not computed as they required upwards of 10 hours for each pairwise registration.

The significant reduction of computation time makes complex real-time registration a possibility which is often sought after in clinical situations. A comparison of mean registration times across the three datasets is outlined in Table (4.4).

4.4 Summary

In this chapter, we discussed a deep learning model for discontinuity-preserving image registration. The proposed model follows an unsupervised learning approach that predicts a displacement field based on two given images. The model can be viewed as some function $g_\theta(I_F, I_M)$ controlled by parameters θ. During training, values of θ are tuned until g_θ is able to produce displacement fields with discontinuity-preserving properties.

To enforce this, a custom regularizer was designed by computing the outer products of displacement vectors of neighboring voxels. An efficient implementation of the regularizer is presented that leverages the parallel processing capabilities of modern GPUs.

Once the model is trained sufficiently, the registration process simply involves evaluating g_θ. Depending on the image resolution, this shortened registration times by several orders of magnitudes compared to traditional inverse approaches. Our model is able to capture local motion discontinuous where expected, while performing competitively against existing state-of-the-art methods in terms of quantitative metrics. Furthermore, we show that employing a learning-based approach is much more time efficient as registration time only requires seconds as opposed to minutes and/or hours. However, learning-based methods suffer a setback where the training procedure is very time consuming. In addition, a model learns to generalize based on the training data. If two images need to be registered are drastly different compared to the training data, the model will require retraining as the generalization is not easily transferred.

Chapter 5

Conclusion and Future Work

Image registration has been around for several decades and it continues to be a challenging problem as requirements often shift on a case-by-case basis. In particular, the recovery of motion discontinuities remains an ongoing research topic due to the different types of allowed motion. The large variety of feasible movements pose many different types of internal boundary conditions which makes it very difficult, if not impossible, to enforce. Instead, a list of plausible local movements within a small neighborhood was procured, and a model was constructed to encourage these requirements. A secondary objective is to reduce the computation time required to register two images. We achieve this by taking advantage of the recent advances in deep learning methodologies and applying them to the image registration problem.

In this work, we combined the two objectives by presenting an unsupervised learning-based model with a custom regularizer. As discussed in Chapter 4, the regularizer divides an image volume into small image patches. A penalty score is assigned to each image patch by examining local displacement vectors against neighboring displacement vectors. We find that this approach offers a reasonable middle ground compared to fully global penalty scores (e.g. diffusion regularizer) and local penalty scores (e.g. sum of squared differences).

Although the training set was relatively small, our experimental results show that the model was able to detect motion discontinuities while maintaining comparable performance with current state-of-the-art methods. Furthermore, our model significantly reduce computation time by several orders of magnitude, making real-time registration achievable and also allowing successive registration in a relatively short time frame.

A drawback of the proposed model is its sensitivity to noise. In particular, since $\mathcal{L}_{\text{disc}}$ is computed by comparing local displacement vectors with neighboring vectors individually, there are no mechanisms in place to discourage local chaotic behaviors in the displacement field. A possible remedy is to treat all displacement vectors within an image patch as a collective unit then quantifying the motion changes along adjacent image patches, similar to the Viscek model of collective motion [44]. Our model can be extended to include additional information, such as segmentation masks and edge information. By leveraging known prior information, locations with image discontinuities are provided rather than predicted solely from image intensities. A two-stage approach can also be taken, where the task of predicting image discontinuities is fully offloaded into the initial stage. The prediction from the initial stage can then be passed into our proposed model for registration. This will require further optimization of the network architecture and the loss functions, as including an additional stage will increase the memory and computation requirements significantly.

A driving motivation of this work is to head towards a fast all-purpose discontinuity-preserving image registration model. One question often asked was whether such a model even exists? On one hand, we simplify the requirements by specifically defining the desired behavior in our model. On the other hand, do these requirements reflect real-world physical movements? And if not, is there a common ground between the two extremes? Although the answers to these questions remain a mystery, the model shown in this thesis provides a promising first step towards finding these answers.

Appendix A

Proof of Representer Theorem

Proof. Let $\phi(x) = k(x, \cdot)$. Suppose that f is a minimizer, we then write f as

$$f(\cdot) = \sum_{i=1}^{n} c_i \phi(x_i) + v$$

where v is the component of f in the orthogonal complement of $\text{span}(\phi(x_i))$. Since k is a reproducing kernel, then evaluating f at x_j gives,

$$\begin{aligned} f(x_j) &= \left\langle \sum_{i=1}^{n} c_i \phi(x_i) + v, \phi(x_j) \right\rangle \\ &= \left\langle \sum_{i=1}^{n} c_i \phi(x_i), \phi(x_j) \right\rangle + \langle v, \phi(x_j) \rangle \\ &= \left\langle \sum_{i=1}^{n} c_i \phi(x_i), \phi(x_j) \right\rangle \\ &= \sum_{i=1}^{n} c_i \langle \phi(x_i), \phi(x_j) \rangle. \end{aligned}$$

Note that the above is independent of v, therefore the loss function $E = E[x_i, y_i, f]$ must also be independent of v. For $g(\|f\|_{\mathcal{H}})$, we have

$$\begin{aligned} g(\|f\|_{\mathcal{H}}) &= g\left(\left\|\sum_{i=1}^{n} a_i\phi(x_i) + v\right\|_{\mathcal{H}}\right) \\ &= g\left(\sqrt{\left\|\sum_{i=1}^{n} c_i\phi(x_i)\right\|_{\mathcal{H}}^2 + \|v\|_{\mathcal{H}}^2}\right) \\ &\geq g\left(\sqrt{\left\|\sum_{i=1}^{n} c_i\phi(x_i)\right\|_{\mathcal{H}}^2}\right) \\ &= g\left(\left\|\sum_{i=1}^{n} c_i\phi(x_i)\right\|_{\mathcal{H}}\right) \end{aligned}$$

with the inequality holding because g is strictly increasing, and equality holding if and only if $v = 0$. Therefore any minimizer must require $v = 0$. Hence a minimizer must be in the form

$$f^*(\cdot) = \sum_{i=1}^{n} c_i\phi(x_i) = \sum_{i=1}^{n} c_i k(x_i, \cdot).$$

□

Appendix B

Implementation of Loss Functions

Algorithm 1 Local Normalized Cross Correlation Between Images I, J.

Input: Images I, J, kernel size n

Output: $1 - \text{LNCC}$

1: Defining convolution kernel for local sums/means

a: $K \leftarrow \text{ones}[n][n][n]$

2: Compute Local Sums

a: $\mathbf{I} \leftarrow I * K$

b: $\mathbf{J} \leftarrow J * K$

c: $\mathbf{I}^2 \leftarrow (I \odot I) * K$

d: $\mathbf{J}^2 \leftarrow (J \odot J) * K$

e: $\mathbf{IJ} \leftarrow (I \odot J) * K)$

3: Compute Local Means

a: $\boldsymbol{\mu}_I \leftarrow \mathbf{I}/n^3$

b: $\boldsymbol{\mu}_J \leftarrow \mathbf{J}/n^3$

4: Local Normalized Cross Correlation

a: $\mathbf{CC} \leftarrow \mathbf{IJ} - \boldsymbol{\mu}_J \odot \mathbf{I} - \boldsymbol{\mu}_I \odot \mathbf{J} + n^3(\boldsymbol{\mu}_I \odot \boldsymbol{\mu}_J)$

b: $\mathbf{I}_{var} \leftarrow \mathbf{I}^2 - 2\boldsymbol{\mu}_I \odot \mathbf{I} + n^3(\boldsymbol{\mu}_I \odot \boldsymbol{\mu}_I)$

c: $\mathbf{J}_{var} \leftarrow \mathbf{J}^2 - 2\boldsymbol{\mu}_J \odot \mathbf{J} + n^3(\boldsymbol{\mu}_J \odot \boldsymbol{\mu}_J)$

5: $\text{LNCC} \leftarrow \text{mean}\left(\frac{(\mathbf{CC})^2}{\mathbf{I}_{var}\mathbf{J}_{var}}\right)$

6: **return** $1 - LNCC$

Algorithm 2 Computing the Discontinuous Loss

Input: Displacement Field $\mathbf{u}$ ($D \times H \times W \times 3$), kernel size s, weight function w

Output: $\mathcal{L}_{disc}$

distWeight $\leftarrow$ emptylist

$K \leftarrow \text{zeros}[s, s, s, s^3]$

for all $i, j, k = 0, \ldots, s$ **do** ▷ Define distance kernel K

  $K[i, j, k, i + j + k] \leftarrow 1$

  distWeights.append($w(\sqrt{(i - \lfloor s/2 \rfloor)^2 + (j - \lfloor s/2 \rfloor)^2 + (k - \lfloor s/2 \rfloor)^2})$)

end for

for $i = 1$ **to** 3 **do** ▷ Extract neighboring displacement (weighted)

  $\mathbf{u}' \leftarrow \mathbf{u}[:, :, :, i] * K$

  if $i = 1$ **then**

    $\mathbf{u}_{nbhd} \leftarrow \text{expand\_dim}(\mathbf{u}', 0)$

  else ▷ $\mathbf{u}_{nbhd}$ has dimensions $3 \times D \times H \times W \times s^3$

    $\mathbf{u}_{nbhd} \leftarrow \text{Concatenate}(\mathbf{u}_{nbhd}, \text{expand\_dim}(\mathbf{u}', 0))$ along index 0

  end if

end for

for $i = 1$ **to** s^3 **do**

  if $i = 1$ **then** ▷ $\|\cdot\|^2$ along index 0, mean is taken over all voxels

    $P \leftarrow \mathbf{u}_{nbhd}[:, :, :, :, i] \times \mathbf{u}$ ▷ $\times$ is taken along index 0

    $\mathcal{L}_{disc} \leftarrow \text{mean}\left(\frac{1}{2}\log(1 + \|P\|^2) \cdot distWeight[i]\right)$

  else

    $\mathcal{L}_{disc} \leftarrow \mathcal{L}_{disc} + \text{mean}\left(\frac{1}{2}\log(1 + \|P\|^2) \cdot distWeight[i]\right)$

  end if

end for

return $\mathcal{L}_{disc}/s^3$

Appendix C

Additional Experimental Results

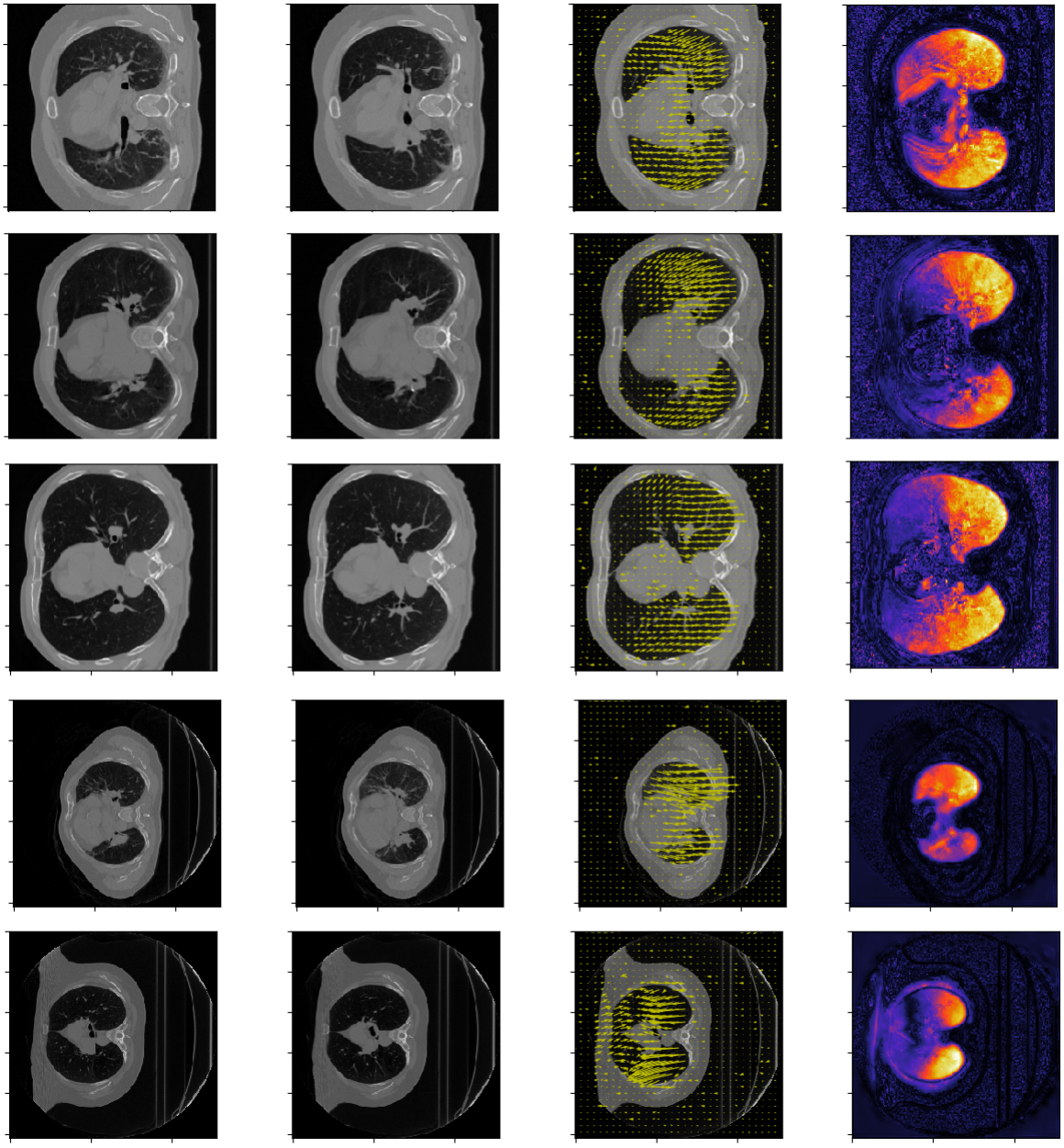

Figure C.1: Additional results of axial slices. From left to right: Fixed image I_F, moving image I_M, registered image $I_M \circ \phi$ with displacement field $\mathbf{u}$ overlap, heatmap of displacement field magnitude.

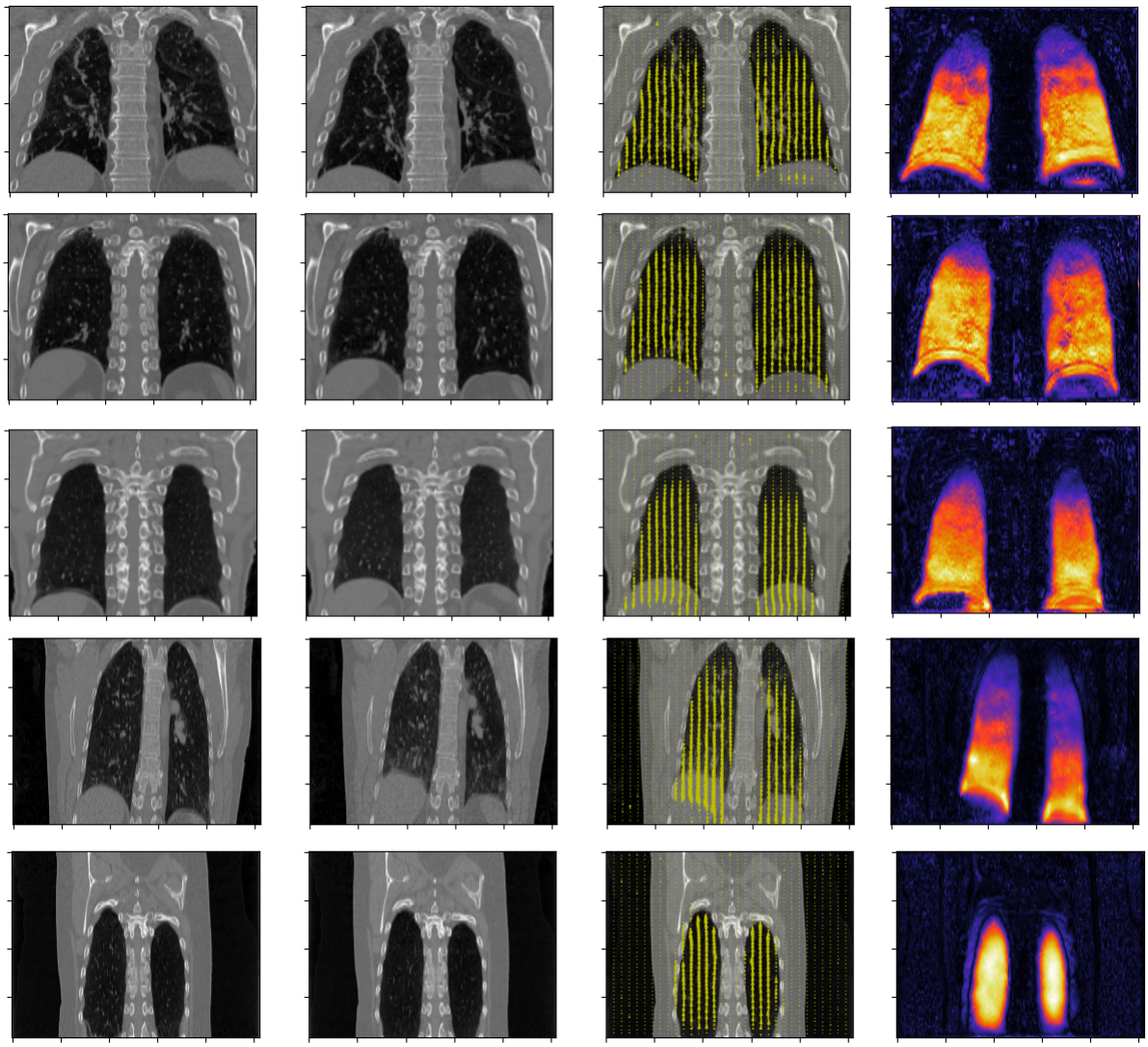

Figure C.2: Additional results of sagittal slices. From left to right: Fixed image I_F, moving image I_M, registered image $I_M \circ \phi$ with displacement field $\mathbf{u}$ overlap, heatmap of displacement field magnitude.

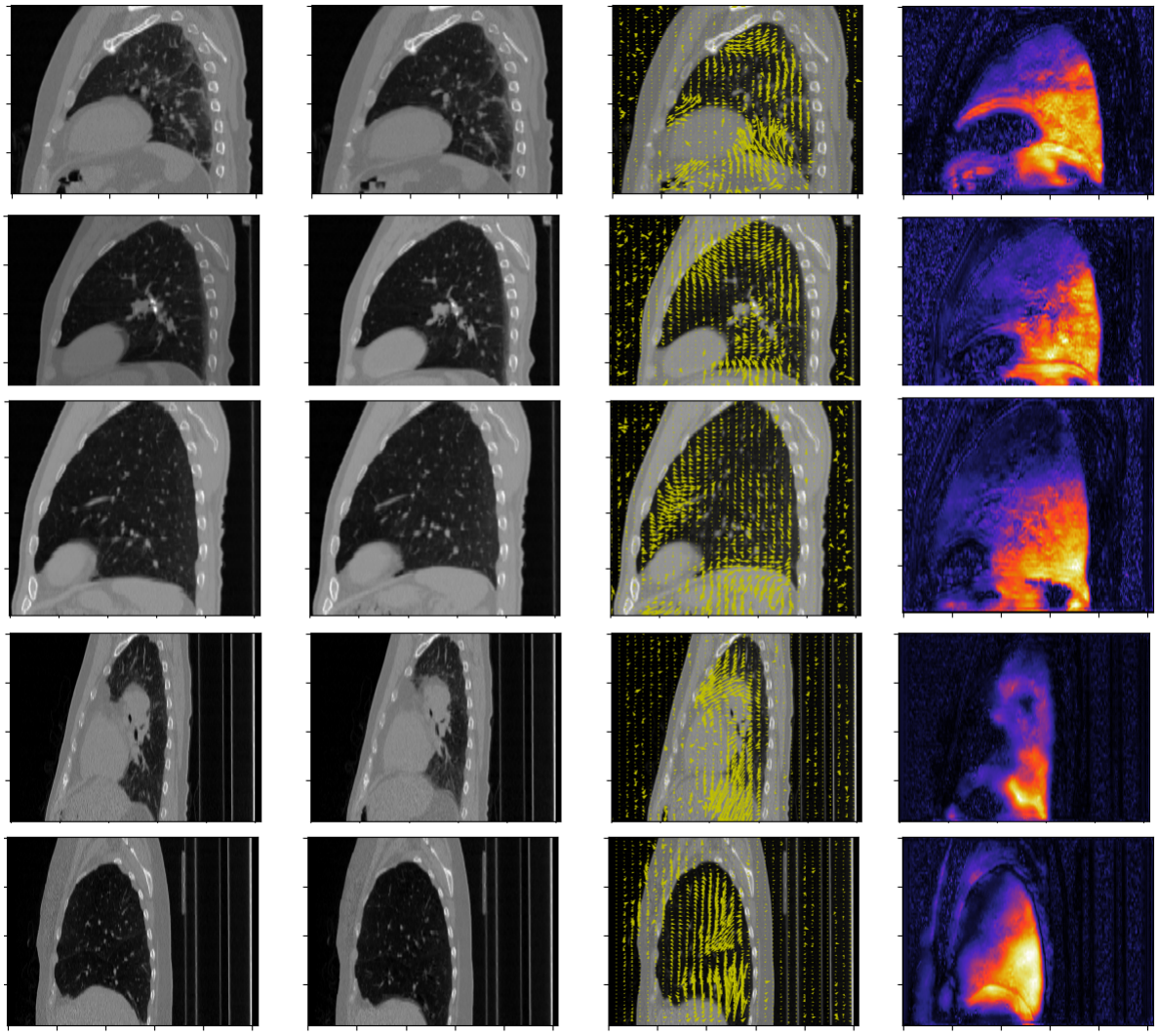

Figure C.3: Additional results of coronal slices. From left to right: Fixed image I_F, moving image I_M, registered image $I_M \circ \phi$ with displacement field $\mathbf{u}$ overlap, heatmap of displacement field magnitude.

Bibliography

[1] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: application to breast mr images," *IEEE transactions on medical imaging*, vol. 18, no. 8, pp. 712–721, 1999.

[2] V. Vishnevskiy, T. Gass, G. Szekely, C. Tanner, and O. Goksel, "Isotropic total variation regularization of displacements in parametric image registration," *IEEE transactions on medical imaging*, vol. 36, no. 2, pp. 385–395, 2016.

[3] C. Jud, N. Mori, and P. C. Cattin, "Sparse kernel machines for discontinuous registration and nonstationary regularization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 9–16, 2016.

[4] R. Sandkühler, C. Jud, S. Andermatt, and P. C. Cattin, "Airlab: Autograd image registration laboratory," *arXiv preprint arXiv:1806.09907*, 2018.

[5] E. Castillo, R. Castillo, J. Martinez, M. Shenoy, and T. Guerrero, "Four-dimensional deformable image registration using trajectory modeling," *Physics in Medicine & Biology*, vol. 55, no. 1, p. 305, 2009.

[6] R. Castillo, E. Castillo, R. Guerra, V. E. Johnson, T. McPhail, A. K. Garg, and T. Guerrero, "A framework for evaluation of deformable image registration spatial accuracy using large landmark point sets," *Physics in Medicine & Biology*, vol. 54, no. 7, p. 1849, 2009.

[7] J. Vandemeulebroucke, S. Rit, J. Kybic, P. Clarysse, and D. Sarrut, "Spatiotemporal motion estimation for respiratory-correlated imaging of the lungs," *Medical physics*, vol. 38, no. 1, pp. 166–178, 2011.

[8] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[9] M. Sonka and J. M. Fitzpatrick, "Handbook of medical imaging(volume 2, medical image processing and analysis)," SPIE- The international society for optical engineering, 2000.

[10] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[11] J. Perry *et al.*, "A class of conjugate gradient algorithms with a two-step variable-metric memory," *Discussion Papers*, vol. 269, 1977.

[12] J. Hadamard, "Sur les problèmes aux dérivées partielles et leur signification physique," *Princeton University Bulletin*, vol. 13, pp. 49–52, 1902.

[13] E. Haber and J. Modersitzki, "Intensity gradient based registration and fusion of multi-modal images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 726–733, Springer, 2006.

[14] J. Modersitzki, *FAIR: flexible algorithms for image registration*, vol. 6. Siam, 2009.

[15] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[16] C. Broit, "Optimal registration of deformed images," 1981.

[17] J. Duchon, "Splines minimizing rotation-invariant semi-norms in sobolev spaces," in *Constructive theory of functions of several variables*, pp. 85–100, Springer, 1977.

[18] J. Salencon, *Handbook of continuum mechanics: general concepts, thermoelasticity*. Springer Science & Business Media, 2001.

[19] G. Wahba, *Spline models for observational data*, vol. 59. Siam, 1990.

[20] E. W. Cheney and W. A. Light, *A course in approximation theory*, vol. 101. American Mathematical Society, 2009.

[21] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, pp. 1171–1220, 2008.

[22] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[23] C. Jud, N. Möri, B. Bitterli, and P. C. Cattin, "Bilateral regularization in reproducing kernel hilbert spaces for discontinuity preserving image registration," in *International Workshop on Machine Learning in Medical Imaging*, pp. 10–17, Springer, 2016.

[24] C. Jud, A. Giger, R. Sandkühler, and P. C. Cattin, "A localized statistical motion model as a reproducing kernel for non-rigid image registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 261–269, Springer, 2017.

[25] C. Jud, R. Sandkühler, N. Möri, and P. C. Cattin, "Directional averages for motion segmentation in discontinuity preserving image registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 249–256, Springer, 2017.

[26] F. Riesz, "Sur une espèce de géométrie analytique des systèmes de fonctions sommables," *Comptes rendus de l'Académie des Sciences*, vol. 144, pp. 1409–1411, 1907.

[27] H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Advances in computational Mathematics*, vol. 4, no. 1, pp. 389–396, 1995.

[28] C. Fox, *An introduction to the calculus of variations*. Courier Dover Publications, 1987.

[29] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[30] D. C. Paquin, D. Levy, E. Schreibmann, and L. Xing, "Multiscale image registration," *Mathematical biosciences and engineering*, vol. 3, no. 2, p. 389, 2006.

[31] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[32] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[33] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[35] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[36] F. F. Berendsen, A. N. Kotte, M. A. Viergever, and J. P. Pluim, "Registration of organs with sliding interfaces and changing topologies," in *Medical Imaging 2014: Image Processing*, vol. 9034, p. 90340E, International Society for Optics and Photonics, 2014.

[37] V. Delmon, S. Rit, R. Pinho, and D. Sarrut, "Registration of sliding objects using direction dependent b-splines decomposition," *Physics in Medicine & Biology*, vol. 58, no. 5, p. 1303, 2013.

[38] A. Schmidt-Richberg, R. Werner, H. Handels, and J. Ehrhardt, "Estimation of slipping organ motion by registration with direction-dependent regularization," *Medical image analysis*, vol. 16, no. 1, pp. 150–159, 2012.

[39] Z. Wu, E. Rietzel, V. Boldea, D. Sarrut, and G. C. Sharp, "Evaluation of deformable registration of patient lung 4dct with subanatomical region segmentations," *Medical physics*, vol. 35, no. 2, pp. 775–781, 2008.

[40] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "Voxelmorph: a learning framework for deformable medical image registration," *IEEE transactions on medical imaging*, 2019.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[42] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 30, p. 3, 2013.

[43] C. R. Maurer Jr, J. J. McCrory, and J. M. Fitzpatrick, "Estimation of accuracy in localizing externally attached markers in multimodal volume head images," in *Medical Imaging 1993: Image Processing*, vol. 1898, pp. 43–54, International Society for Optics and Photonics, 1993.

[44] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical review letters*, vol. 75, no. 6, p. 1226, 1995.